Dynamic Reduction of Rollbacks in Wireless Multi-user Virtual Environments

Abdul Malik Khan[1], Sophie Chabridon[1], and Antoine Beugnard[2]

[1] Institut TELECOM, TELECOM SudParis
CNRS UMR SAMOVAR
9 rue Charles Fourier
91011 Evry cedex, France
{Abdul_malik.Khan,Sophie.Chabridon}@institut-telecom.fr
[2] Institut TELECOM, TELECOM Bretagne
Computer Science Department, CS83818
29238 Brest cedex 3, France
Antoine.Beugnard@institut-telecom.fr

Abstract. In distributed virtual environments such as multiplayer games, where many users interact in real time while communicating through a network, the users may have an inconsistent view of the game world because of the communication delays across the network. Consistency maintenance algorithms must be used to have a uniform view of the game world. The majority of these algorithms use rollback mechanisms to correct the inconsistencies that occur because of the disorder of the arrival of update messages. These rollbacks are very costly, especially when playing a game, using high-latency wireless networks, on mobile terminals which have limited memory and processing speed. In this paper, we present a dynamic and adaptive approach for reducing the number of rollbacks in distributed virtual environments on wireless mobile devices. This approach takes into account the underlying network latency and the semantics of the game virtual world to dynamically decide whether a rollback is needed in case inconsistencies have occurred or can be possibly avoided. We evaluate our approach on a simplified version of a Football game on hand-held devices and show that this dynamic rollbacks' reduction approach improves the responsiveness of the game and maintains consistency of the game state while limiting the use of processing power and memory space.

Keywords: Multiplayer Mobile Games, Latency Hiding, Data Synchronization, Consistency Algorithm.

1 Introduction

Networked Multiplayer Games are becoming more and more popular with the advances in hardware technologies and enriched game design improving immersive feelings. However one of the main issues hindering the real-time interaction in these games is network delays. To enforce strong consistency, events transfer

should satisfy total order [2], however this would degrade [4] the performance of the game. Because of the unordered delivery of events and the network delays induced by unreliable protocols, the game state can be inconsistent at different points in time.
To address this issue, synchronization algorithms are used to reach a consistent state at all the players. There are two approaches to state consistency maintenance. In the conservative approach, all the participants must wait for the acknowledgement of their update messages and reach a global consistent state before advancing. Unfortunately, this approach cannot be applied to continuous applications such as multiplayer games where the state of the simulation changes not only as the result of user actions but also with the passage of time. Also, waiting for acknowledgements would violate the interactivity property. In the optimistic approach, the participants process events without waiting for the arrival of late events and repair any potential inconsistency when it actually occurs. This approach is suitable for mobile continuous interactive applications as it preserves the game experience of the players even with a high network latency and variations in the connectivity as in wireless networks.

Different mechanisms [10,3,5,6,13] can be used to repair inconsistencies in the optimistic approach. Among the proposed solutions, rollback mechanisms are a popular solution to remove the inconsistencies that occur because of late arriving messages when using unreliable network protocols. However, these rollback mechanisms use static approaches to solve inconsistencies without keeping in view the changes in the network and game environments. In wireless networks, the communication delays are higher as compared to wired networks and also jitters can occur impacting the network delays. Also, we believe that in a multiplayer game, different objects and regions in the game have different consistency requirements that vary during the game play as we have already discussed in [9]. In this paper, we propose an optimistic approach based on a dynamic rollback mechanism which adapts its behaviour according to the changes in the network conditions and the consistency requirements of the game at a particular time. The aim is to considerably reduce the number of rollbacks which occur because of variations in the wireless network latency and to improve the playability of the game on mobile devices. Furthermore, we introduce the idea of critical actions to denote highly sensitive events in the game which impact the outcome of the game results. The update messages representing these actions cannot be discarded and must be eventually delivered. Also the rolling back of these messages can have an adverse effect on the result and the usability of game. Our approach tries to avoid, whenever possible, the rollbacks of such critical actions (or events) by relaxing the consistency requirement during the game play.

This paper is organized as follows. In section 2, we discuss some related works. In section 3, we discuss the need for a more dynamic approach to consistency maintenance. In section 4, we discuss our proposed solution for dynamic rollback reduction. In section 4.1, we propose the idea of critical actions and then in section 4.2 we use this concept to define relations between game events that can help discard late arriving events without performing any rollback. In section

5, we present an example scenario to explain our approach. Based upon the concepts discussed in sections 3 and 4, we give in section 6 an algorithm for the reduction of the number of dynamic rollbacks. In section 7, we discuss the trade-off between interactivity and consistency. In section 8, we give the results of our experimentation on the proposed approach. Section 9 concludes this paper and discusses some future work.

2 Related Works

Time Warp [10] is an optimistic synchronization algorithm that allows remote participants to execute their events without the guarantee of a causally consistent execution. Time Warp takes a snapshot of the state at the reception of a command and issues a rollback to an earlier state if a command older than the last executed command is received. With an unreliable transmission protocol and high delays in wireless networks, the number of late arriving commands can greatly increase the number of rollbacks which can be very costly for users of mobile terminals keeping in view their limited memory and processing capacity. Moreover, frequent rollbacks can affect the playability of the game. The local lag approach can be used with Time Warp to force the local events to wait for the late arriving events before displaying them. This allows to reduce the number of inconsistencies that can occur, but at the cost of a decreased responsiveness of the game.
In [5,6], the authors present an event correlation algorithm for mirrored server architectures to decrease the number of rollbacks. The events which are not correlated to earlier events and arrive late can be discarded without rolling back the previously executed commands. These papers, however, do not present any definition or approach for finding the correlation between events.
[13] uses a correlation algorithm for P2P Massively Multiplayer Online Games (MMOGs). According to their definition, two events are correlated if "they both update the same state variables associated to a given game element". Apart from repairing inconsistencies, an approach called dead-reckoning [1] can be used to predict the future positions of objects in the game world for increasing the interactivity of the virtual environment and hiding the network latency. Additionally, the local lag [10] allows to delay the display of local command(s) so that update messages from remote players can arrive, thereby avoiding a causal disorder of messages. We have already proposed in [9] the idea of dynamically changing the parameters for dead-reckoning and local lag according to the network delays and the positions and speeds of the game's objects. In this paper, we combine this idea with our dynamic rollback reduction approach to achieve consistency in multiplayer games according to the need of the situation.

3 A Dynamic Approach to Consistency Maintenance

In the previous section, we discussed different synchronization algorithms used for consistency maintenance in multiplayer games. Because of the rollback and re-processing of commands, Time Warp is very costly in terms of memory and processor usage and hence is not very suitable for mobile games. The local lag approach, combined with dead-reckoning, is suitable for high latency networks. But because of changing delays and jitters in wireless networks, a fixed local lag can cause inconsistencies in the game state across different nodes. Also, we believe that the message discarding approach presented in [7] can be interesting to combine with local-lag. This is because in wireless networks, there are messages which arrive late due to the network jitter and hence must be discarded as these late arriving messages may cause inconsistencies. In this section, we present a dynamic approach in which the consistency maintenance algorithm changes its parameters according to different factors of the environment: e.g. the network load, the type of objects, the location of an object in the virtual world etc. In this section, we first discuss the conditions under which these different parameters are dynamically changed and then we combine these different approaches into the form of an algorithm.

3.1 Observations

While playing a multiplayer game, some inconsistencies may occur due to the communication delays across the network. The game programmers estimate these delays in order to compensate for the late arriving messages from remote users. Because of the jitters, especially in wireless networks, these delays may vary greatly. Therefore, it is necessary to observe the network load during the game play and to compensate for these delays accordingly. In a rich multiplayer game, there are many different types of objects in the virtual environment. The velocities and directions of these objects vary according to their nature. For example, in a tennis game, the speed of the tennis ball is greater than the speed of the players. Also, a player has to react sharply to the movement of a ball. Therefore, these different types of objects in the game world have different consistency requirements. The algorithm responsible for consistency management has to react not only to the varying latencies of the underlying network infrastructure, but also has to deal appropriately with the various types of objects of the game. We also observe that the consistency requirement for an object not only depends upon its speed, but also upon its position in the virtual world. For example, in a car racing game, we need strict consistency management when two cars are very close to each other and they both are approaching the finishing line. Based upon these observations, we believe that a consistency maintenance algorithm must take into account the context of the game along with the variations in the network communication delays.

3.2 An Adaptable Local Lag

When a message about an object is received from a remote user, this object has a certain distance, possibly zero, from the other objects, destination, or any other important entity in the game world, called pivot. We therefore change the value of the local lag according to three factors:

1. If the object concerned by the message we have received from the remote user is coming closer towards the pivot, we reduce the value of the local lag. If the object is going away from the pivot, we increase the value of the local lag upto a certain limit called *Local-lag*$_u$. This increase can be continuous with respect to the motion of the object, or can be discrete based on zones as in [12]. The rate of the change of the value of the local lag is application dependent, and the programmer must specify it during the development of the game. In the next section, we present an approach to help the programmer to specify these values to a component responsible for consistency management.
2. The value of the local lag also changes according to the network load. When the number of messages arriving later than a certain waiting time reaches a certain level N_d, we increase the value of the local lag. This increase in the number of messages arriving late can be due to jitters in the network. Note that the value of the local lag is proportional to the network latency, but we cannot increase it more than a certain limit because it would have a bad effect on the responsiveness. This upper limit can be dependent on the pace of the game. If the game or an object in the game demand high responsiveness, we should not increase the local lag value above a certain limit. This limit can be specified by the game developer.
3. We can set different local lag values for different types of objects according to their importance in the game. For example, we can have a smaller value of local lag for the ball and a higher value for the players in a tennis game. Indeed, the responsiveness of the ball must be high to satisfy the interaction requirement as in [14]. Again, we need an interface provided by the component implementing the algorithm so that game developer can specify the relative values of local lags for different objects in the virtual environment.

3.3 Adaptable Dead Reckoning

The dead-reckoning approach relies on a threshold represented the maximum distance tolerated between the real object trajectory and the predicted one. This threshold value should be situation and environment dependent. Different objects have different consistency requirements at different regions in the game and hence the dead-reckoning should adapt dynamically according to the situation. For this purpose, we propose the idea of critical regions to denote those regions where strong consistency is required.

Critical Regions. We define a Critical Region as a region in the game where we need strict consistency so that all the players have a consistent view of the game in that region. A critical region is one in which inconsistencies can violate the fairness of the game or can annoy a player because his expectations are violated. Therefore, we propose to use real update messages instead of predictions in those regions. For example, in the case of a tennis match, if the ball hit by one player is touching the ground near the base line on the other side of the court, and the opponent player

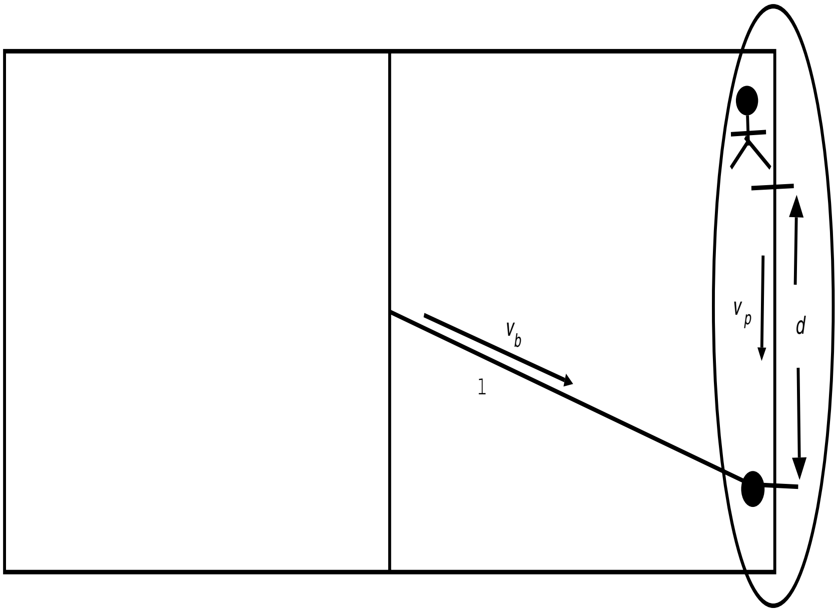

Fig. 1. Area around the base line constitute a critical region in Tennis game

is quite far from the ball, we stop the dead-reckoning so as to increase the fairness between the players. The ball will touch the ground only when the original message is received and hence the result of the score will be correct.

The calculation of the decision whether to use dead-reckoning or not can normally be done through some easy arithmetics. For example, in Figure 1, let v_b and v_p be the current speed of the ball and the maximum speed of the opponent player respectively. Let l be the distance covered by the ball from the centre of the court to the point where it touches the ground and d be the distance of the player from that point. The ball will reach the ground in l/v_b time units and during that time the player can cover a distance of $v_p * l/v_b$ distance units. So if $d > v_p * l/v_b$, then the player cannot reach the ball, and we can stop dead-reckoning from near the net where we did the calculation. Of course, the ball will stop and jump into the air for a short period of time near the centre of the court. This will not affect the playability and the fairness of the game because we know that the opponent could not reach the ball in any case. Note that, if we do not use the idea of critical region and continue with dead-reckoning, we may predict a wrong position for the ball touching the ground near the base line and that wrong decision may cause one player to loose a point which he otherwise would have won.

Apart from critical regions, the dead-reckoning mechanisms is also dependent upon the network load and the nature of the object. For example, we can have different dead-reckoning thresholds for different objects according to their movement and importance in the game world. This threshold must be specified by the game developer to the component responsible for consistency maintenance.

4 Dynamic Reduction of the Number of Rollbacks

In this section, we present the concepts that can help to dynamically reduce the number of rollbacks in high latency wireless networks.

4.1 Critical Actions

We first introduce the concept of *critical action* (or *event*). We define *critical actions* as commands in the game, that unlike position update messages, affect the output of the game for other objects. For example, a shoot command can increase the points of one object and can kill or reduce the moving capability of another object. Hence, we consider the shooting command as a critical action. The delivery of critical actions is highly essential, however, as we will see in the next section, there are times when dropping such events does not violate the outcome of the game. Moreover, the rollback of a critical action can cause the users to quit the game because their expectations were violated [14]. This happens when there is difference between the state that they could achieve because of their own actions and the resultant state after the rollback. Therefore, efforts must be made to deliver the critical actions before users' expectations are violated and rollbacks of critical actions should be avoided whenever possible. We argue that in certain cases, such as with a very high latency in wireless networks and when the action is not taking place in a critical region, the critical action message can eventually be dropped if arriving *very* late. The definition of *very* depends upon the nature of the game and the value of the network latency and the local lag at that time.

4.2 Weak and Critical Correlation

Introduced in [5], the concept of obsolescence states that an event that arrives late at a recipient while some event that was issued earlier than this event (i.e. having greater time stamp) has already been processed, is obsolete and must be discarded. Additionally, the concept of correlation states that if this obsolete message is correlated to an earlier processed event, then all the events till that correlated event must be rollbacked and reprocessed along with this new arriving message. [5], however, does not define any mechanism to calculate the correlation between any two events. [13] defines two events to be correlated if they are associated with a single object.

We introduce here the new concepts of *weak correlation* and *critical correlation* by incorporating the notions of *critical regions* and *critical actions*. We propose that any late arriving event should be considered obsolete if it is neither in a critical region nor it is a critical action. A critical action in a critical region should never be considered obsolete and must be guaranteed to be eventually delivered.

We now give the definitions [1] of the properties of weak and critical correlations based upon the concept of correlation.

Correlation χ,
Given two events e_{ci} and $e_{cj} \in E_c$, where E_c is the set of all events that belong to o_i and $o_j \epsilon O$, the set of all objects, with time stamps T_i and T_j, such that $T_i < T_j$, then,
$e_{ci} \chi e_{cj}$ iff
$(e_{ci}; e_{cj}, \mathrm{s}) \rightarrow s1 \wedge (e_{cj}; e_{ci}, \mathrm{s}) \rightarrow s2 \wedge s1 \neq s2$. $s1$,$s2 \in S$, the set of all states.

It means that the ordered execution of correlated events is necessary to reach a consistent state. We now define the two properties of weak and critical correlations.

Property 1: *weak correlation* χ_w Two events are weakly correlated if they are correlated to each other, but are either non-critical actions or do not occur in any critical zone.

Property 2: *critical correlation* χ_c Two events are critically correlated if they are correlated to each other, and in addition, those two events belong to the set of critical actions and they occur in a critical zone.

All correlated events that are non-critical events are weakly correlated.

4.3 Relaxed Consistency

In our definition of relaxed consistency, when the number of late arriving messages increases above a threshold value, thereby indicating a high network latency, we rollback only those events that are critically correlated and we discard weakly correlated events. We also increase the local lag value for critical events so as to further decrease the number of rollbacks and better satisfy the user expectations with regard to the game.

Whenever the number of late arriving messages decreases below the threshold value, then we stop the discarding of weakly correlated events. This is because the number of rollbacks has already decreased following the decrease in the network latency, so processing weakly correlated events will not increase the number of rollbacks beyond a certain limit. Also we decrease the value of the local lag to increase interactivity for the users. For all other events that are not critical, we discard them when arriving late, because it will not affect the playability of the game and will unnecessarily increase the number of rollbacks thereby wasting the limited processing power and memory space of mobile terminals.

[1] We adopt a simplified version of a syntax based on a Plotkin-style operational semantics (Plotkin 1981 [11]). In particular, $(e_i; e_j, s) \rightarrow s*$ denotes an initial game state s from which the final game state $s*$ is reached through two subsequent events, namely e_i and e_j.

5 An Example Scenario

In figure 2, we show an example game scenario in which a character hits a moving target (an animal in the figure) with their gun shots. To hit the target, the shooter first sends a warning (or ready) sign before shooting it. Without a warning sign, it cannot shoot the target. In the figure, a circle around the animal denotes a critical region where a gun shot can hit the target.

Suppose that the warning sign $w2$ arrives later than the actual shot $s2$. Since $s2$ hits an area which is outside of the critical region, we do not need to perform any rollback when $w2$ arrives since it will not affect the outcome of the game. We say that $s2$ and $w2$ are weakly correlated. In case $w1$ arrives later than $s1$, then we have to apply a rollback because we cannot discard W1 as it will violate the game rule of warning before shooting. We say that $w1$ and $s1$ are critically correlated. If we have already experienced a large number of rollbacks thus suggesting a high network latency, we increase the value of the local lag so as to give more time to late arriving messages and hence decrease the number of rollbacks in the critical region.

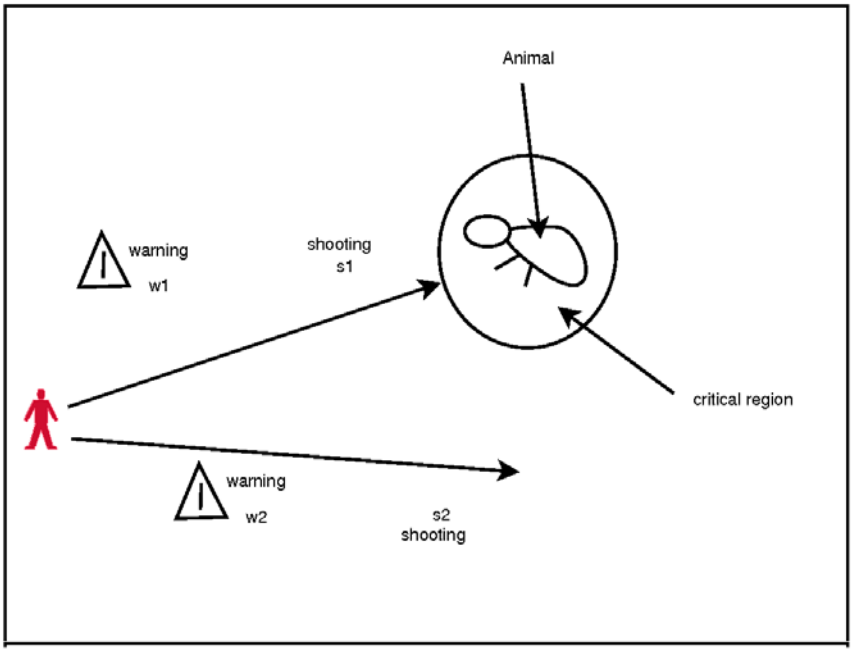

Fig. 2. A game with two players

6 Dynamic Rollbacks Reduction Algorithm

In this section, we present a dynamically adaptive synchronization algorithm based on the concepts of obsolescence and correlation using the optimistic approach as already discussed in the previous sections. In our algorithm, we propose to minimize the number of rollbacks with our dynamic approach. The purpose of reducing the number of rollbacks is to avoid unexpected behaviour during the game and to allow a smooth flow of the game. Also, in case of a game played on mobile phones, rollbacks are very costly in terms of their use of memory and processing resources which are limited on small portable devices.

In our approach, we combine the use of local lag and the notions of critical region and critical action. A local lag is the artificial introduction of a delay (both at emitter and receiver nodes) so as to allow an ordered processing of those messages which were isssued earlier but arrived later than other messages because of the network delay. If the number of rollbacks oversteps a certain threshold, we increase the value of the local lag, so that more and more messages could arrive on time. However, when a player enters a critical region and the latency is not very high, we reduce the value of local lag and increase the frequency of message sending to achieve high interactivity. When the number of required rollbacks increases in the critical region, then, instead of doing rollbacks for all arriving events, we discard weakly correlated events and increase the frequency of update messages to minimize the dependency, i.e. correlation, on a single late arriving message.

The pseudo code for the algorithm is given in Algorithm 1. In the given algorithm, lines 10 through 14 are the most important as far as the concept of correlation and obsolescence and the avoidance of rollbacks are concerned.

We discard any message that arrives late (and became obsolete) and is not correlated with any previous message (lines 7 and 8). Otherwise, if a message arrives late but is correlated to some previous message e_c and the number of rollbacks is less than a certain threshold, then we apply a rollback on all the messages processed before e_c including e_c itself and re-process them in the correct order (line 11). In line 14, we rollback only critically correlated events as now the latency is very high and the number of rollbacks has reached a certain limit. In lines 16 to 18, we increase the value of the local lag if the number of rollbacks oversteps some limit so as to avoid processing related messages in an incorrect order. This will also decrease the number of obsolete discarded messages which could be related to any future message(s). We keep a different value for rollbacks threshold in critical regions and change the values of the local lag and Dead-Reckoning thresholds in these regions if the number of rollbacks reaches a certain level (lines 24 and 25). Lines 2 to 25 are repeated in a loop throughout the execution of a game session.

There is no doubt that the interactivity will be decreased at the cost of consistency and correctness of results. We discuss the issue of interactivity in the next section.

Algorithm 1. Correlation and Obsolescence based adaptive algorithm

```
1: Calculate the local lag at the beginning of the game for each class of objects
   according to network latency and responsiveness requirement of the object(s).
2: Change the local lag if the network load has changed or the location of an object
   is changed
3: if the message arrives during its local-lag specified time then
4:    Buffer the message according to its local lag value before playing out
5:    GOTO line 20 (apply DR)
6: end if
7: if the message is obsolete (not arriving in its speicified local lag time) and not
   correlated with events already processed during the local lag time interval then
8:    Discard the message
9: else
10:    if the number of rollbacks is less than a certain threshold then
11:       Rollback the messages, process this message and then all others
12:    end if
13: else
14:    Rollback only critically correlated events and discard all others
15: end if
16: Calculate the number of rollbacks
17: if  the number of rollbacks reaches a certain limit then
18:    Increase the value of local lag.
19: end if
20: Apply Dead Reckoning
21: if the object has entered the critical region and/or the network load has changed
    then
22:    Change the threshold value for DR for that object
23: end if
24: if  the number of rollbacks reaches a certain limit in the critical region then
25:    decrease the value of DR threshold and increase local lag value
26: end if
```

7 Responsiveness vs Consistency

By waiting for late arriving messages with an increase of the local lag value, the consistency is improved. However, it means that even local actions (and those remote events that arrived earlier because of low latency) must be delayed before playing out. Thus, it decreases the responsiveness (or interactivity) of the game. If a game requires high responsiveness, then we need to reduce the value of the local lag which can disturb the causal order of events (and increase the need for rollbacks), thus compromising the consistency of the system. Hence there is a trade-off between these two properties in a distributed interactive application.

We propose to apply different degrees of interactivity and responsiveness for different situations and/or regions of a game. For example, if a player has to shoot a static target, we need high consistency but low responsiveness since the target is static and during the time the bullet hits (or misses) the target, we do not need high interactivity from the system, but we need a fair result.

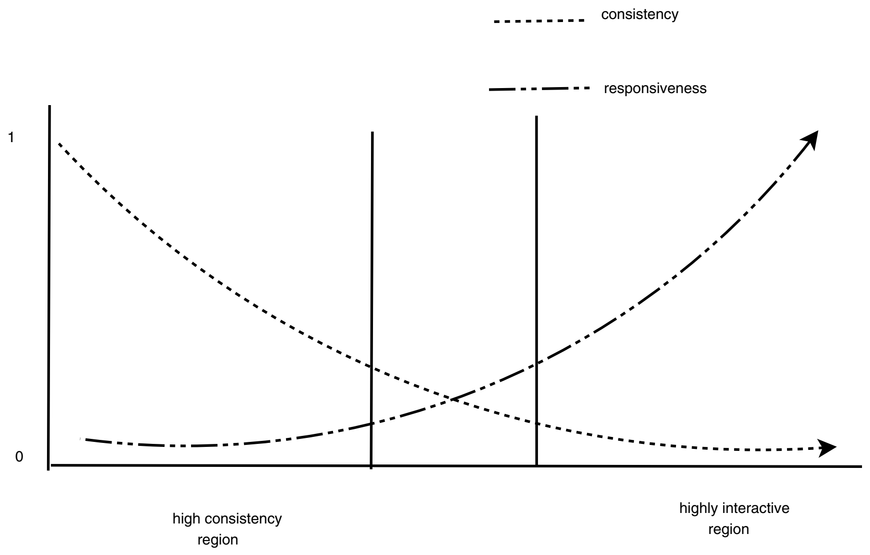

Fig. 3. Trade-off between consistency and responsiveness in different game regions

By applying a suitable value of local lag, the shooting player will observe that his shoot action has taken place at a slow pace, but will get the true results. However, we need high responsiveness from the system in some other cases, such as hitting a ball coming towards a player in a baseball game. The trade-off between responsiveness and consistency is shown in figure 3. Although we have shown consistency (and responsiveness) on a scale from 0 to 1, where 1 means absolute consistency, absolute consistency is never achieved in distributed virtual environments and hence consistency should be compromised for the sake of a high system responsiveness.

8 Evaluation

In the evaluation of the proposed approach, we are interested primarily in the measurement of two parameters.

1. the number of rollbacks;
2. the amount of dropped events;

For experimenting our proposed solution, we have developed a simple Football (Soccer) game using J2ME and Java servlet technologies. The game logic resides on a mobile phone and different players interact with each other via a server which is a simple message queue servlet. In the game, we have characters representing the different players and a goal post, which corresponds to a critical region. Each player has a circle of specific radius representing the critical region around them. A player can earn points by either hitting another player or the

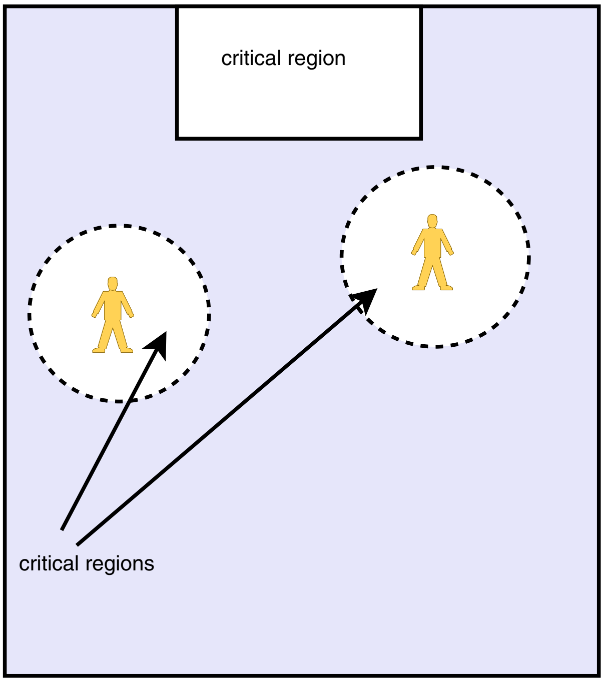

Fig. 4. A simple Football game

goal post with a bullet. The rules of the game allow the player to be hit only in the critical region shown by the rectangular shape. The game is shown on Figure 4.

The server randomly selects messages to be delayed deliberately by storing them in a server side queue before clients can receive these messages. When a delayed message arrives at the client, it calculates whether this message is correlated with another message or not. In our game semantics, a message is correlated with another message if it belongs to a bullet (critical action) or if a player entered a critical region. In case of a very high consistency, we can drop even the *hit* message because it will have no effect upon the result since our game rules allow a player to be hit only in a critical region. If the message is not correlated, we simply drop it, otherwise we apply a rollback mechanism and reprocess all the already processed messages. We continuously calculate the number of rollbacks and apply our dynamically adaptive algorithms by changing dead-Reckoning and local lag thresholds to control the number of rollbacks. In essence, we measure the number of rollbacks in the critical regions and outside the critical regions, and the number of messages discarded while the players are in a critical region and outside it. We also compare these results with the fixed-correlation based approach and the Time Warp algorithm, where there is no concept of obsolescence and correlation. The results are shown in figure 5.

From the figure, it is clear that the number of rollbacks required for Time Warp is higher than for the other two approaches. This was expected, as Time

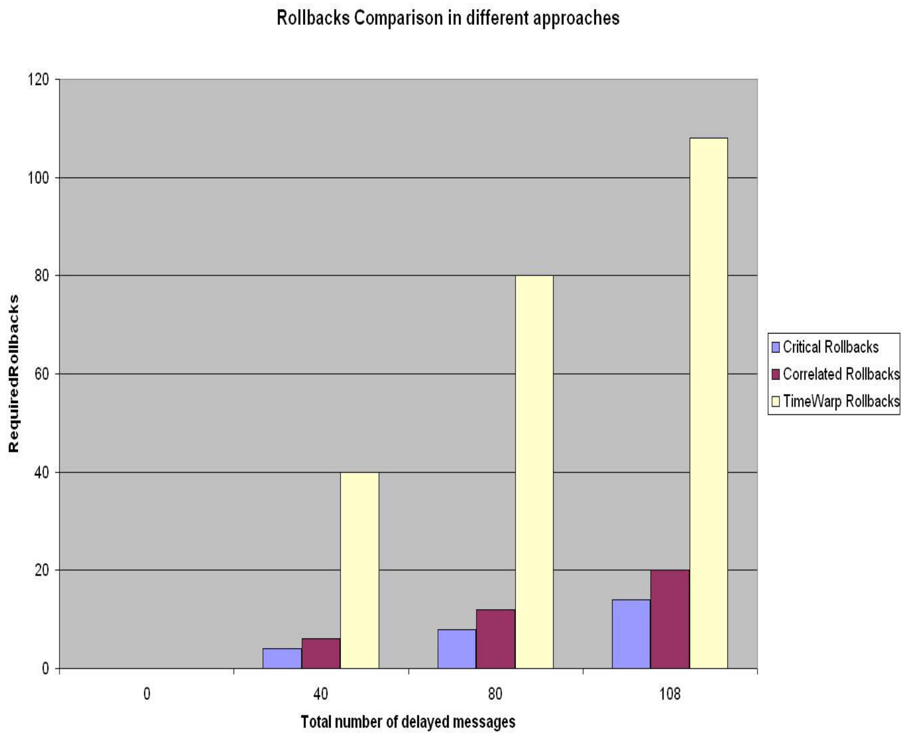

Fig. 5. Rollbacks comparison in three different approaches

Warp does not drop any message and applies rollbacks for all late arriving messages. The result of our approach is better than the result for fixed-correlation based approach, because we rollback only those messages which are critically important, and discard non-critical messages in case of high delays.

Figure 6 compares the number of messages re-processed per rollback as a function of their correlation probability. In case of a low probability of correlation (high No-Correlation probability on the x-axis), our Critical Correlation based approach reprocesses a smaller number of messages as compared to the Time Warp and Simple Correlation based approaches. This is because in case of lesser correlation probability, there are even lesser chances that the messages will be critically correlated and hence they are discarded without affecting the outcome of the game. However, in the case of a high correlation (low No-Correlation probability percentage), our result approaches that of the Time Warp and Simple Correlation based approaches. This happens only when almost all messages are critically correlated which is very rare in a real game scenario, where players enter and exit the critical regions and only a few of their actions are critical.

Figure 7 compares the increase in the total number of rollbacks as a function of the time elapsed during the game play.

Note that on the Y-axis, we show the number of rollbacks required as a whole i.e. when the rollback mechanism has to be applied, and not the number of

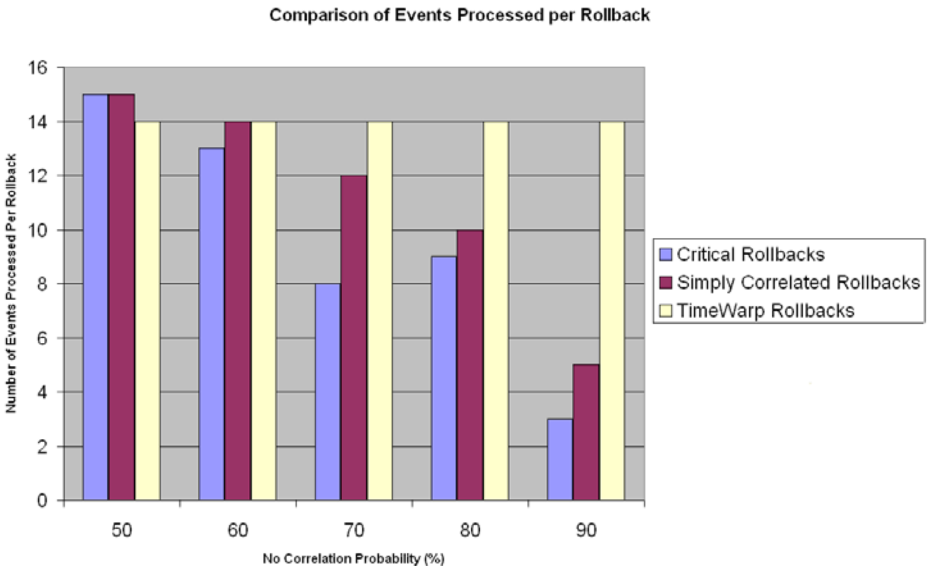

Fig. 6. Comparison of events processed per rollback

messages to be rollbacked which can be manifold higher than these numbers. For the reasons discussed in the previous paragraph, our approach performs better than simple correlation. The increase in the number of required rollbacks is linear in case of simple correlation because we periodically delay only correlated update messages at regular intervals. In the figure, for the critical correlation-based approach, the increase in the number of rollbacks is not uniform (the graph is not straight). This is because even if we delay messages periodically at regular intervals, they may not be critically correlated at that junction of time and hence are discarded without the need for applying a rollback mechanism. It must be noted here that our approach is dependent upon the strategy of the players of the game as when and where they send critically correlated messages. In the worst case scenario, when all the received messages are critically correlated, our mechanism is equal to that of the simple correlation mechanism.

9 Conclusions and Perspectives

In this paper, we introduced a dynamic and adaptive approach for consistency maintenance. We proposed a dynamic mechanism for the reduction of the number of rollbacks in multiplayer games on mobile phones. We introduced the concepts of critical actions and critical regions, and with the help of these two notions, we showed that we can relax the consistency requirements whenever the game rules allow it, thus reducing the number of rollbacks considerably. We showed, through our experimentation on a simple Football game, that our approach performs better than the Time Warp and the simple correlation approaches.

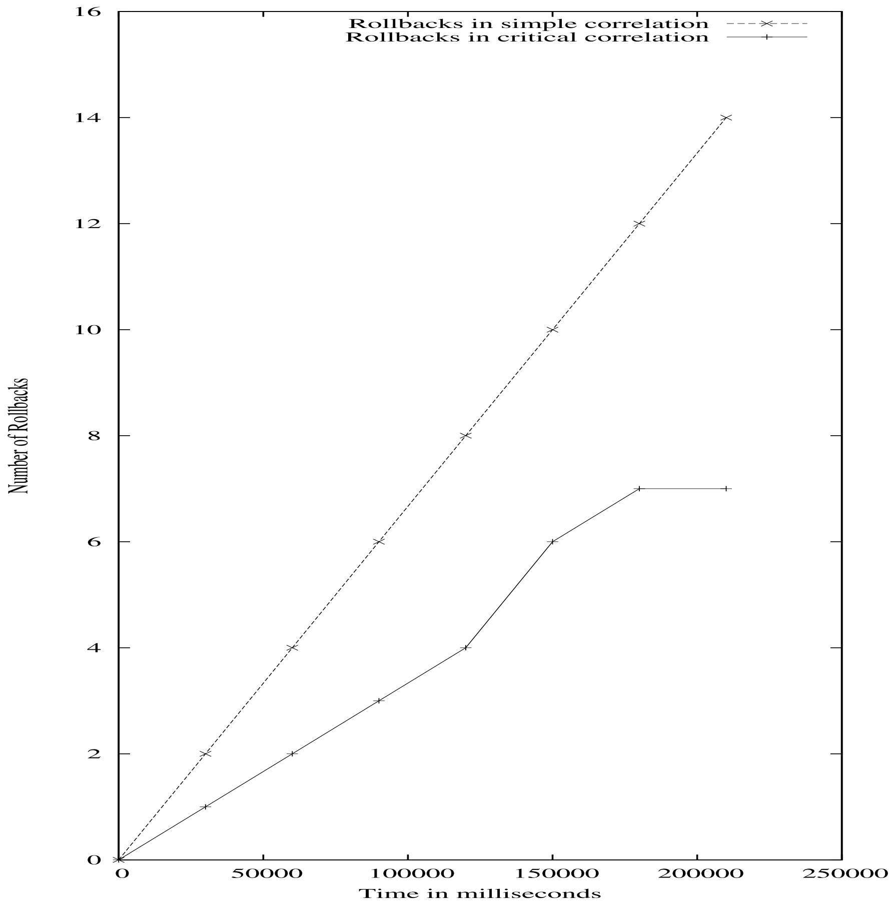

Fig. 7. Rollbacks comparison as a function of elapsed time

In the future, we would like to insert this algorithm as a part of our synchronization medium, a distributed component responsible for consistency maintenance as proposed in [8]. This way, the game developer(s) will be able to reuse these complex consistency maintenance algorithms without changing the game logic code.

Acknowledgments. This work is in parts supported by the Higher Education Commission of Pakistan (www.hec.gov.pk).

References

1. Application protocols. In: IEEE Standard for Distributed interactive Simulation. IEEE Std. 1278.1-1995 (1995)

2. Cheriton, D.R., Skeen, D.: Understanding the limitations of causally and totally ordered communication. SIGOPS Oper. Syst. Rev. 27(5), 44–57 (1993)
3. Cronin, E., Filstrup, B., Kurc, A.R., Jamin, S.: An efficient synchronization mechanism for mirrored game architectures. In: NetGames 2002: Proceedings of the 1st Workshop on Network and System Support for Games, pp. 67–73. ACM Press, New York (2002)
4. Défago, X., Schiper, A., Urbán, P.: Total order broadcast and multicast algorithms: Taxonomy and survey. ACM Computing Surveys 36(4), 372–421 (2004)
5. Ferretti, S., Roccetti, M.: A novel obsolescence-based approach to event delivery synchronization in multiplayer games. Int. J. Intell. Games & Simulation 3(1), 7–19 (2004)
6. Ferretti, S., Roccetti, M.: Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games. In: ACE 2005: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, pp. 405–412. ACM, New York (2005)
7. Ishibashi, Y., Tasaka, S., Tachibana, Y.: Adaptive causality and media synchronization control for networked multimedia applications. In: IEEE International Conference on Communications (ICC), pp. 232–241. IEEE Computer Society (2001)
8. Khan, A.M., Chabridon, S., Beugnard, A.: Synchronization medium: a consistency maintenance component for mobile multiplayer games. In: NetGames 2007: Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games, pp. 99–104. ACM, New York (2007)
9. Malik Khan, A., Chabridon, S., Beugnard, A.: A dynamic approach to consistency management for mobile multiplayer games. In: NOTERE 2008: Proceedings of the 8th International Conference on New Technologies in Distributed Systems, pp. 1–6. ACM, New York (2008)
10. Mauve, M., Vogel, J., Hilt, V., Effelsberg, W.: Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. IEEE Transactions on Multimedia 6(1), 47–57 (2004)
11. Plotkin, G.D.: A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus (1981)
12. Santos, N., Veiga, L., Ferreira, P.: Vector-Field Consistency for Ad-Hoc Gaming. In: Cerqueira, R., Campbell, R.H. (eds.) Middleware 2007. LNCS, vol. 4834, pp. 80–100. Springer, Heidelberg (2007)
13. Xiang-bin, S., Fang, L., Ling, D., Xing-hai, Z.: An event correlation synchronization algorithm for mmog. In: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2007, June 30-August 1, vol. 1, pp. 746–751 (2007)
14. Zhou, S., Shen, H.: A consistency model for highly interactive multi-player online games. In: ANSS 2007: Proceedings of the 40th Annual Simulation Symposium, pp. 318–323. IEEE Computer Society, Washington, DC (2007)