Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2008, Article ID 642314, 9 pages
doi:10.1155/2008/642314

Research Article

The Playing Session: Enhanced Playability for Mobile Gamers in Massive Metaverses

S. Cacciaguerra and G. D'Angelo

Department of Computer Science, University of Bologna, Mura A. Zamboni 7, 40127 Bologna, Italy

Correspondence should be addressed to G. D'Angelo, gda@cs.unibo.it

Received 31 January 2008; Revised 14 April 2008; Accepted 23 May 2008

Recommended by Jouni Smed

Internet ubiquity and the success of mobile gaming devices are increasing the interest in wireless access to virtual environments. Mainly due to the mobility factor and wireless medium features, traditional gaming architectures are not enough to guarantee good levels of playability and fairness to mobile gamers. We suggest a new mechanism, called playing session, capable of controlling communications between mobile devices and the game infrastructure. In case of network failures, a mimicking mechanism is in charge of playing, until the communication channel is restored. The goal is to reproduce, with an adequate level of mimesis, the user behavior. According to this approach, it will be possible to enhance the overall playability of Internet games without requiring any modification to the existing communication infrastructure.

1. INTRODUCTION

"*You take the blue pill, the story ends here, you wake up and believe whatever you want to believe. You take the red pill you stay in wonderland and I'll show you just how deep the rabbit hole goes.*" In this way, Morpheus offers Neo to be woken up by an illusory simulated reality, called Matrix, which is developed by intelligent machines in order to use human beings as their source of energy. Matrix is just one of the many visions describing the future Internet as a global cyberspace humans can explore and shape through their avatars. Words like Cyberspace, Metaverse, and Matrix are synonymies: all of them refer to a virtual reality-based evolution of Internet. In this scenario, the avatar is a tool allowing humans to interact with a metaverse (i.e., a meta multiuniverse). Over the last years, the interest of the gaming industry has led to the implementation of many metaverses called virtual worlds (VWs). Many of them are so realistic that they have an economy, government, and currency of their own (e.g., World of Warcraft [1], Second Life [2], Project Entropia [3], Sociolotron [4]). Thanks to the massive diffusion of wireless Internet access and to the increasing miniaturization of hardware devices, there is a growing interest in extending the massive online gaming to also nomadic users. Due to the unreliable nature of the wireless medium and to the mobility, this kind of gamers would require special mechanisms to maintain a good level of playability, at least from a technological point of view. In this work, we propose a new mechanism aiming at enhancing playability for all gamers (both wired and wireless). Our mechanism introduces a new level in the communication protocol stack that is in charge of controlling communications between clients and servers. The mobile nature of wireless gamers can be often the cause of interruptions and lags in the communications between devices and gaming infrastructure. We propose the playing session (PS) mechanism to enhance the gaming playability while maintaining good levels of equity and fairness between all users. The main task of PS is to monitor gaming actions and to quickly react in case of network failures, hence taking control of all avatars disconnected from the players (i.e., orphan avatars). The PS scope is not limited to network failures, it is triggered every time the gamer is unable to fulfill the deadline set by the game progress (i.e., hard-to-use input hardware interface of small devices or user disabilities) [5].

The reminder of the paper is organized as follows. Section 2 illustrates the problems when participating in massive metaverses from mobile devices. Section 3 introduces related works. Section 4 highlights the main design issues to enhance the gamers' playability. Section 5 describes the proposed system architecture. Section 6 presents a case study: a clone of the Armagetron game. Finally, Section 7 concludes this work with some final remarks.

2. BACKGROUND

As seen in the intoduction, the wireless access to VW could be a big market success. It is a common assumption that, in the next years, Internet will be ubiquitously available, at least in some parts of the world. Furthermore, due to many practical and cost-related reasons, the last hop will be often based on wireless technologies. The combined effect of a widespread availability of Internet and the development of a new generation of wireless devices will lead to a massive amount of gamers interested in VW.

Today's portable devices integrate wireless network technologies into high-performance multimedia terminals, with the explicit aim at enabling distributed multiplayer gaming [6]. The potential of these devices is very high, but it is influenced by the characteristics of the underlying network technologies. For example, the playability of real-time multiplayer games is dominated by the end-to-end network latency [7]. In the case of 802.11 WLAN networks (Wi-Fi), clients should be able to communicate with an acceptable latency and data rate, while many problems are due to the movement of users. For example, what happens when a nomadic user wants to participate in an Internet game, but he is too far from all access points (APs)? In this scenario, packets coming from the mobile device and directed to the VW servers might be lost due to the lack of connectivity in the area or delayed due to the interaccess point handoff. The wireless scenario might generate many different situations: *horizontal handoffs* (it refers to the process of transferring a data session from one channel to another), *vertical handoffs* (it refers to a change in the technology, e.g., from Wi-Fi to UMTS), *interferences* (generating transmission errors), *closure* of the communication channel (e.g., deauthentication and disassociation). Moreover, many communication protocols were designed to comply with static nodes. What happens when a mobile device connects to an AP that belongs to a different internet service provider (ISP)? In this case, the mobile device should be able to obtain a new valid IP address. This could lead to an incorrect management of the network communications, and even, in a positive case, to the reconfiguration, while resume mechanisms might require many seconds to bring the system back to a working state. In order of importance, we classify the effects of the wireless communication faults: *short interruptions* due to the loss of some data (i.e., loss of packets, datagrams, or segments), *long interruptions* mainly due to the reconfiguration and the resume activities (i.e., protocol disconnections or application shutdowns), *permanent interruptions* mainly due to incorrect communication management at the application level (i.e., unexpected shutdown of the application, system crash). In this scenario, users playing from a mobile device and therefore using unreliable networks could be severely disadvantaged with respect to "wired gamers." They might lose some match turns (which is very unfair) and be also disconnected from the whole system. In all of these cases, the overall playability would be dramatically decreased.

3. RELATED WORK

A lot of studies focus on the communication-related problems arising from the distributed nature of the gaming architectures, both from the server and the client point of view [8]. That is, the impact of packet loss and communication latency on the playability and the fairness of Internet games have been widely investigated. Beigbeder et al. [9] have studied the effects of the packet loss and latency on user performance in "Unreal Tournament, 2003." In this case, the analysis is focused on, the so-called, first person shooters (FPSs), a class of games that is considered more sensitive than others to the changes in network performances. Instead, in [10] the effect of latency on users' performance has been inspected in case of a real-time strategy (RTS) game: Warcraft III. As expected, due to their nature, RTS games can tolerate a limited amount (less than a second) of latency without impacting on the overall outcome. Conversely, FPSs are greatly affected by latency: even a modest increase of the communication latency reflects in a deep degradation of the user performance. For the sake of simplicity, the proposed solutions can be divided into two main approaches:

(i) solutions requiring some kind of support from the network layer (i.e., quality-of-service-based mechanisms);

(ii) solutions based on mechanisms totally independent from the network layer's guarantees and assumptions.

Following approach (i) in [11], a quality-of-service (QoS) extension has been proposed to mobile ad hoc routing in order to support real-time applications. In this case, the main achievement was the reduction of the loss rate, while maintaining acceptable latency and jitter. A radically different approach (ii) is followed by [7] proposing a framework, called Rendezvous, based on an optimistic synchronization scheme that provides a consistency mechanism for high-latency environments. A more general approach can be found in [8]: in this work different mechanisms are introduced and analyzed, to deal with inconsistencies due to the distributed nature of the gaming architecture. In our opinion, at least in a foreseeable future, the majority of wireless networks will not provide any form of QoS to real-time applications. The PS mechanism falls within the approach previously defined as (ii). Upon that premise, it is worth noting that a QoS-enabled network layer would be complementary to the proposed mechanism.

4. ON DESIGNING A PLAYER SESSION

We define playability as the user's satisfaction while playing. In other words, this means that the gameplay quality is related to the "fun to play" and the "usability," with particular attention to the responsiveness and the sensation of a realistic participation. Moreover, the playability is also related to other typical aspects of a game, such as: storyline quality, customizability, control, intricacy, strategy, and the realism's degree. In our case, we are only interested in the responsiveness and immersive sensations because they are directly related to the communication performance. On the contrary, other aspects (e.g., the storyline quality) can be considered as features of both the game and its "mechanics."

In this scope, we designed and implemented a new mechanism called playing session (PS) that is in charge of monitoring the client-server communication channel. The mechanism will be triggered by network failures, mainly due to the gamers' movement. The core of the PS mechanism is a new level in the communication stack (see Figure 1). More in detail, the PS is composed by two parts: the participatory framework (PF), in charge of detecting and reacting to the network failures and the mimicking mechanism (MM) that tries to reproduce, with an adequate level of mimesis, the user behavior. In our architecture, the PF is present on both sides (client and server). On the mobile device (i.e., the client in our gaming architecture), the PF is able to detect if the currently used game server is unreachable or reachable with a certain delay. On the game server, the PF is able to detect whether the application on the mobile device is experiencing problems (i.e., the movement of the player has caused a network failure). In this case, the PF gives the control of the player's avatar to the mimicking mechanism (MM) that will be in charge of playing, until the communication channel is restored. In this way, the game can continue the progress avoiding that the whole system is affected by the fault of a single player. On the other side, it is worth noting that a disconnected gamer will not be able to continue to play: he will not be able to play until the connection is restored again.

4.1. *Interactivity*

The gamers expect to play fluently, without taking care of problems deriving from devices' limitations (e.g., size of the control keys and screen, battery duration) or due to the unreliability of communications [5]. We believe that players should be able to maintain their gaming style independently from the network failures. For example, a stronger player should never loose with a weaker one only because a network failure reduces his ability to interact with the VW. Our idea is to relax the temporal constraints of the game progress up to the sensorial perceptivity threshold. This means that the length of a single turn is upper bounded by this threshold. In this way, it is possible to use all the available time to wait for "delayed moves" (i.e., events delayed due to network failures). We should keep in mind that losing an action, with low frequency, is not so critical neither for the game system nor for the player. In case of missing moves, efficient predictive techniques exist (e.g., dead reckoning mechanisms) [12]. Some games are so "fast and furious" that players have not a detailed perception of the whole situation. In this context, it is possible to adaptively decrease or increase the duration of each turn. In detail, the adaptation mechanism could be based on the users' responsiveness.

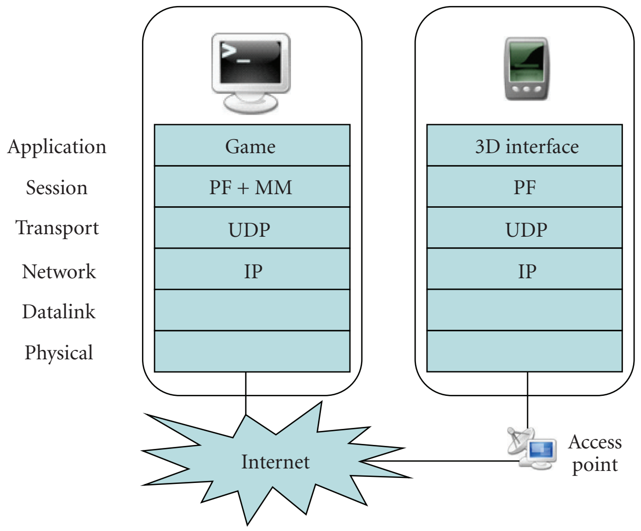

Figure 1: The playing session: a new layer in the communication protocol stack.

In few words, it is mandatory to maintain the temporal order of the moves: it would be possible to decrease or increase their frequency, but always under the sensorial perceptivity threshold. In particular, from the participants' point of view, we do not accept as a solution (aimed at enhancing the playability) the presence of ghost avatars in the scene. In the gaming jargon, the word "ghost" refers to a "frozen" avatar that is not reachable by its gamer.

4.2. *Coherence*

It is really unpleasant, for gamers, to find frozen avatars in the middle of the scene. A common and really unsatisfactory solution to this problem is to set some prefixed actions to support an "orphan avatar" (i.e., an avatar that is disconnected from its gamer). In order to avoid this unpleasant experience, it is important to maintain a good level of coherence inside the game. In this case, we define coherence as uniformity among players: each player should be able to participate in the game in the most correct way, without limitations severely degrading his experience. In a perfect world, the network communications are reliable and effective, the input device has a good level of usability, the operating system does not crash, and the game system does not fail. The real world is very different: network faults are frequent and the usability of the input devices is often unsatisfactory. In this scenario, an MM would be able to increase the level of the game coherence: for example, it could replace the player should his actions not arrive to the game server within the perceptivity threshold. An important assumption is that the mechanism should be able to play at the same level of the substituted gamer. An MM that plays better than the gamer would make pointless his future actions or introduce a new form of cheating. Conversely, an MM worse than the gamer could reduce his chance of victory, generating inadequate actions that are very difficult or impossible to recover by the gamer. An interesting side effect is that the other players should not be able to detect which avatars are controlled by a real gamer and which are controlled by the MM. From a different point of view, it should be impossible to detect which gamers are experiencing network problems by simply looking at the behavior of their avatars. In this connection, the MM would be completely transparent to other players. The research and implementation of this mechanism is out of the scope of our work because is mainly related to artificial intelligence (details can be found in [13]), but some considerations useful to improve the PS effectiveness will be following discussed.

Unfortunately, the believability of an avatar is subjective, since it is influenced by the culture and the skills of the other players [14]. Moreover, in order to show an adequate degree of humanness [15], the avatar should adopt human-like reaction and decision times, avoid to give superhuman capabilities and realize some tactical/strategy reasoning [16]. After these considerations: what is the best action that the MM should play when it is in control of an orphan avatar? In this case, the main point is to define which is the exact meaning of "best action" [13]. There are, at least, two different viewpoints:

(i) the player's viewpoint;

(ii) the viewpoint of the other gamers.

In the former case, the best action would be the most predictable one; in the latter, it would be the action that is able to reproduce the strategy of the user. Furthermore, in (i), the new action should be the natural consequence of the correct progress of the game, and in this case, it could be quite easy to be predicted. This approach is effective only if the gamer loses the control of the avatar for a very limited amount of time, and with low frequency. The main advantage is related to the implementation: it could be based on a simple lookup table of state-action pairs. An interesting example of a related technique is dead reckoning [12]: in this case the prediction of the future state of the avatar is based on the current state (e.g., the future position of the avatar is forecast taking in account its current position, speed, and direction). Unfortunately, a mechanism based on hard-coded default actions is unable to comply with long-term disconnections: the avatar would be quickly recognized as a fake. If most part of the actions are played by the MM or if it is triggered too often, then the avatar will likely start to show nonhuman behaviors. In this case, in order to maintain an acceptable level of coherence, the MM should also take into account other factors, as an example the stochastic/strategic behavior of real gamers. Traditionally, this problem has been solved by increasing the complexity of the algorithms used to control the avatar. These algorithms are not easy to be designed and implemented because many combinations of events and situations have to be considered and some of them are very hard to be predicted in advance. Following this approach, it would be possible to extend some of the dead reckoning concepts. For example, an extended dead reckoning for first person shooters (FPSs) would require supporting many actions such as: jumping, changing the weapon, shooting enemies, and in some cases also more complex actions (e.g., setting a trap). An alternative approach is to raise the level of abstraction to a tactical or strategic level [17]. In practice, it would be possible to monitor each user to infer his typical gaming behavior: the MM would be instructed to follow the strategy of the gamer. Furthermore, the MM should be adaptive and able to capture the real essence of the strategy, instead of a collection of disconnected actions. In this sense, we need techniques [18] capable of analyzing a collection of task pairs (instance, solution) without knowing the dynamics of the solution (i.e., without formalizing the algorithm). With a collection formed by an adequate number of instances, an MM should be able to substitute the player's strategy/ability with an appropriate level of mimesis and with a good level of generalization. Another problem is related to the computational effort required to obtain timely results. The MM should be able to infer a "good action" in a short time: also in this case the amount of available time is bounded by the perceptivity threshold. Finally, different approaches can be followed in the production of the model knowledge. A first approach would be to collect offline the data for instructing the MM. On the other side, a more complex and costly approach would collect data during the game progress (i.e., online). In this case, it would be possible to dynamically adapt the MM mechanism to the strategy evolution and to different gaming events.

4.3. Equity/fairness

Interactivity and coherence are the bases for achieving a good level of equity/fairness in the game. In a distributed system, fairness can be defined as the guarantee to avoid the starving of any process: each process should have the same priority in the access of shared resources. In this case, all processes should have the same chance to progress. In gaming, the aim is to guarantee substantial equity among all players. In a perfect world, each gamer would be allowed to play the same number of actions, with the same frequency of the other participants. Unfortunately, problems due to network communications can have a high impact on the game equity. From our point of view, avatars and gamers should be decoupled, we consider avatars as processes that, in case of network failures, are separate from gamers.

The chance for each gamer to play the same number of actions with the same frequency in the match is a key point to evaluate the game equity, and therefore both aspects should be carefully measured. As a consequence, if the PF is able to promptly detect the network failures and in case of missing events, to substitute the gamer, then it would be possible to ensure fair gaming conditions. It is worth noting that this does not mean that all gamers, at the end of the match, will have played the same exact number of actions. Some gamers could have played fewer actions with respect to others, but in any case, the number of processed actions will be the same for each avatar. In this sense, the PS mechanism gives all gamers the same chance to win the game.

5. SYSTEM ARCHITECTURE

The PS mechanism has been implemented using a Multiagent system (MAS). In detail, the prototype has been integrated in the system for parallel agent discrete event simulation (SPADES) [19], a well-known MAS. On this basis, we added a PS layer in the protocol stack (as shown in Figure 1). As said in Section 4, the PS is in charge of monitoring the communication between the gamer and his avatar and to react in presence of failures or delays. In particular, our system architecture must be able to cope with two different situations:

(i) the loss of a low percentage of actions, with a low frequency (i.e., short interruptions);

(ii) the loss of a substantial percentage of actions (i.e., a train of actions) or low percentage with high frequency (i.e., long or permanent interruptions).

In the first case (i), the PF detects the problem and tries to maintain a good level of interactivity: forcing MM to control the orphaned avatar and hence to produce moves within the perceptivity threshold and with an adequate level of mimesis. In the latter (ii), the PF will try to resume the control of the gamer on the avatar and, in the meantime, MM will produce an adequate strategy to control the avatar. As a consequence, PF (see Figure 2) is composed by a couple of modules, the user participatory framework (UPF) and the avatar participatory framework (APF). The UPF, which is accommodated in the gamer device, checks the state of communications and verifies if the gaming architecture is reachable or not. On the server side, APF monitors the communications with gaming devices.

5.1. *Avatar participatory framework (APF)*

At the beginning of a match, the APF initializes a UDP communication to the UDF. In the meantime, the MM will control the avatar until it starts receiving actions from the gamer. The communication channel is used by the gamer to take the ownership of a specific avatar: usually, the pairing between the gamer and his avatar will last for the whole game duration. The APF tries to maintain active the communication with the corresponding UPF, in case of a long or permanent interruption, it will wait for the recovery of the avatar. When the communication is active, the APF continuously checks two different timeouts. The former (i.e., the action timeout) prevents the slowdown of the game progress due high latency in the interactions between the gamer device and the server. This is implemented by the APF monitoring the responsiveness of the related UPF: the measured latency has to remain under a predefined upper bound (i.e., the perceptivity threshold). Clearly, as above mentioned, if the upper bound is exceeded then the APF forces the MM to play an action in place of his gamer. If the MM plays a number of consecutive times that exceeds a maximum value (defined as transport timeout), then the APF sets the state of the communication as "broken." As a direct consequence, the APF shuts down the existing communication channel and changes its state to "listening" mode, and waits for the recovery of the avatar.

5.2. *User's participatory framework (UPF)*

The UPF (placed in the gaming device) continuously checks whether its avatar is reachable, or if the recovery of the communication is necessary. As described in Section 5.1, the UPF takes the ownership of a specific avatar establishing a communication channel. Every turn, the protocol forces each APF to send a RequestAction event to its UPF. The UPF waits a RequestAction event for a period that is no longer than an action timeout. Furthermore, each RequestAction event is identified by an incremental number. In this way, it is possible to detect if RequestAction events were delayed or lost (i.e., a network failure has occurred). If the RequestAction event comes too late, the UPF buffers the last user-generated action, waiting for the next timeslot. In the meantime, if the player generates another action, then the UPF will overwrite the previously buffered one. In this way, it avoids the delivery of an action that is related to an old state of the game. If the UPF does not receive any event within the transport timeout, then it sets the state of the communication as "broken." In this case, the transport timeout is equal to the maximum number of consecutive action timeouts that can be exceeded: it is worth noting that this value depends on the specific semantics of each game. Finally, if the communication state is set as broken, then the UPF shuts down the existing communication channel and tries to restore the control of its avatar, instancing a new channel.

6. CASE STUDY

In this section, we claim that, even in presence of network failures, the PS mechanism maintains the interactivity within the sensorial perceptivity threshold and does not alter the gamers' strategy. As a consequence, the PS will not alter the progress of the virtual world making the chances of victory of each player unaltered. The PS is completely transparent to the gamer: it is not invasive, it does not affect the satisfaction of the gamer and enhances equity and playability. Its performance measures are strictly related to the following conditions:

(i) C1: the game engine progress respects the sensorial perceptivity threshold;

(ii) C2: each avatar plays the same number of actions, with the same frequency;

(iii) C3: the PS mechanism does not alter the chances to win of each gamer.

If all conditions are met then the PS mechanism is able to maintain a good level of interactivity and coherence, guaranteeing equity between gamers. To support this thesis, a clone of the Armagetron game [20] (inspired by the light-cycles sequence in the Disney movie Tron) has been implemented on top of the PS prototypal implementation. Armagetron is a multiplayer game where participants challenge each other driving a "synthetic motorbike" that leaves behind a wall.

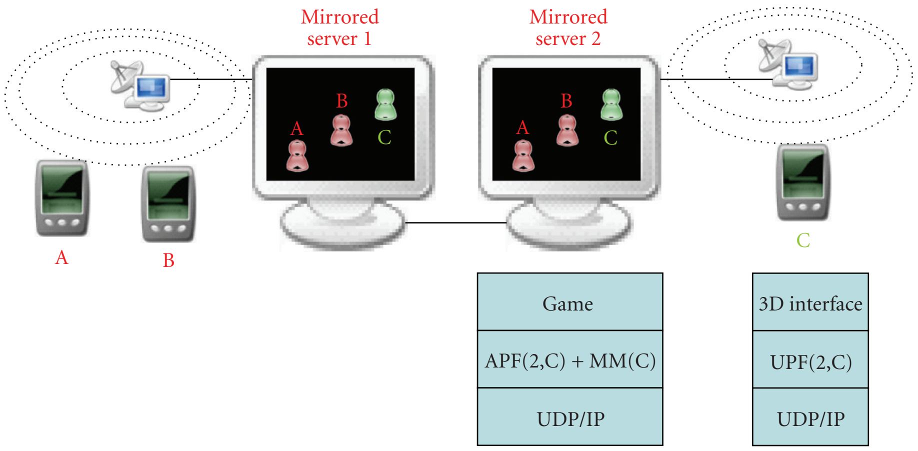

Figure 2: The playing session mechanism.

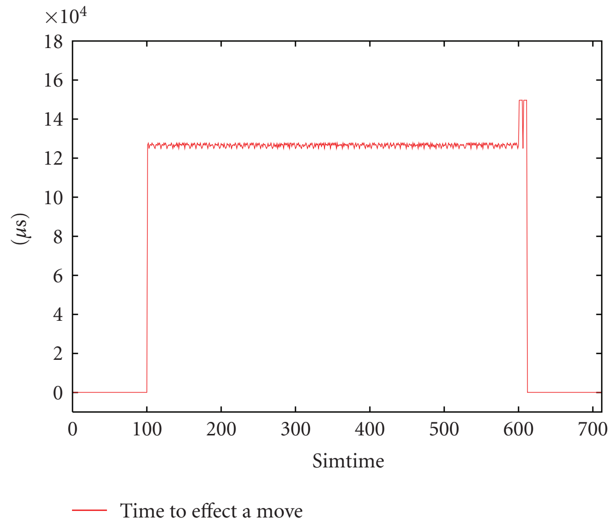

Figure 3: Timeline of the game progress with L = 125 milliseconds and PL = 0%.

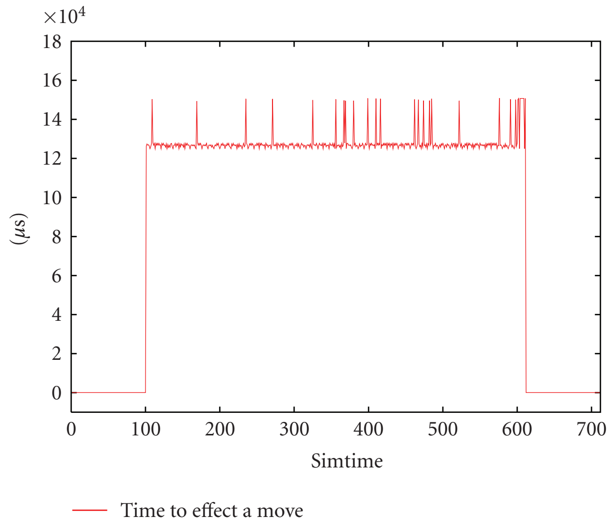

Figure 4: Timeline of the game progress with L = 125 milliseconds and PL = 5%.

During the drive, the motorbikes have to avoid the walls: if a motorbike crashes into a wall or into the borders of the arena then game is over. The aim of the game is to stay alive while killing other player, blocking their path. The winner is the last one alive. To make the game more complex, a motorbike can never stop: it can only accelerate up to the maximum speed and decelerate to the minimum. Turning left or right slows down the speed of the motorbike. Armagetron is interesting because it is a fast-paced multiplayer game, it has a low-complexity implementation but supports sophisticated strategies.

6.1. Network performance

In order to study the performance and the effectiveness of the PS mechanism, we emulated [21] five different network scenarios with increasing packet loss (PL) ratio (0–20%). In this case, the latency (L) was set to 125 milliseconds. We repeated the same matches under different scenarios, in order to study the invasiveness of the proposed mechanisms. In our opinion, the variation of the PL ratio can be used to reproduce the typical network problems of a gamer wandering about the city. The different PL rate should be able to reproduce the following situations:

(i) the gamer is near to the AP and, therefore, the signal strength is very good (i.e., 0% PL);

(ii) the signal is attenuated by obstacles (i.e., 5% and 10% PL);

(iii) the gamer is moving in and out of the coverage area (i.e., 20% PL).

Table 1 reports the average (μ) and the standard deviation (σ) (in microseconds) of the time required to process a turn, the duration of the match (Σ, in microseconds), the percentage of actions played by each gamer with respect to

Table 1: Performance evaluation of PS with different packet losses.

PL	μ (μs)	σ (μs)	Σ (μs)	% played actions (gamer)	# actions (gamer + MM)
0%	126994	3313	64893734	100	500
5%	127950	5633	65382227	96	500
10%	128736	6861	65784246	92.4	500
20%	131010	9289	66946053	82.4	500

Table 2: Evaluation of different strategies.

Strategies	Average occupied space (%)	σ	Average lifetime (simtime)	σ
(I)	6.4	0.8	145	35.9
(II)	6.9	0.7	127	32.1
(III)	4.3	0.4	442	133.1
(IV)	6.5	0.8	561	133.6

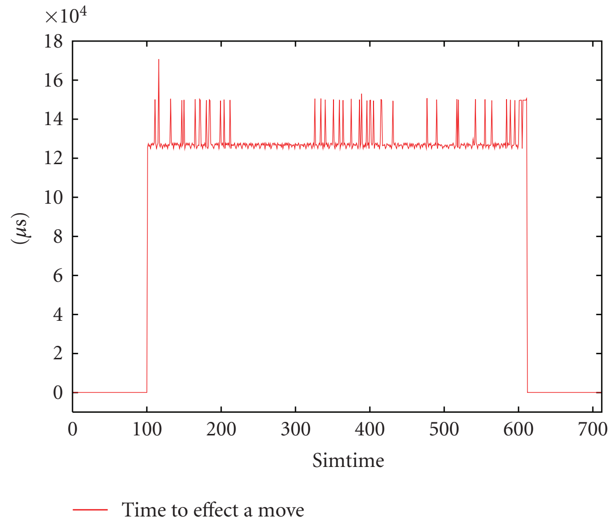

Figure 5: Timeline of the game progress with L = 125 milliseconds and PL = 10%.

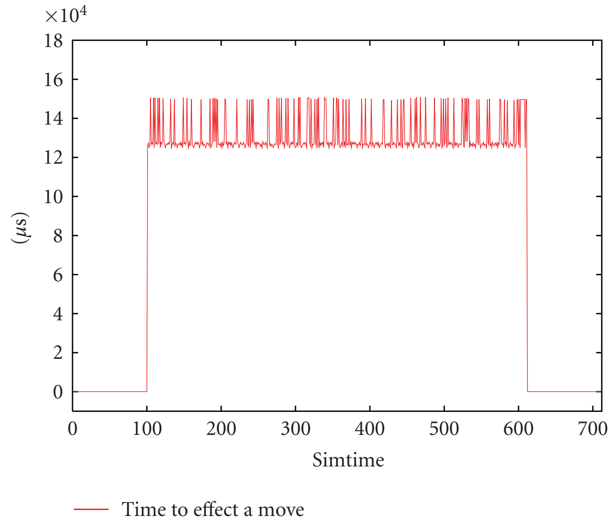

Figure 6: Timeline of the game progress with L = 125 milliseconds and PL = 20%.

the total number of actions played by its avatar, and finally the number of actions played by the related avatar (i.e., the sum of the actions played by the gamer and his MM). Figures 3–6 show a timeline of the game progress: the X-axis represents the simulated time, expressed in turns, (simtime), the Y-axis represents the wall-clock-time required to process a turn (in microseconds). 500 actions are represented in all figures, the game starts at simtime 100 and goes on until the last action is executed. Figures show that the time required for a turn is always under the sensorial perceptivity threshold (150 milliseconds), even in critical situations (i.e., high levels of packet loss). In this sense, the condition C1 is satisfied. Furthermore, the last column in Table 1 shows that the total number of played actions does not depend on the scenario. In detail, the frequency of played actions is comparable, and this is demonstrated by μ, σ, and Σ (see second, third, and fourth columns of the table). In this sense, also the condition C2 is verified.

6.2. *Effectiveness of different gaming strategies*

In order to show that the PS mechanism is not "invasive" (i.e., does not alter gaming outcomes), we have verified how much the results of the same match played in different network scenarios diverge. In our opinion, the chances of victory of a strategy, with respect to another one, should remain unchanged, despite the activation of the PS mechanism. In order to produce an adequate number of trials for the comparisons, we have automated the gaming process, reproducing the most common strategies used in Armagetron. In detail, we created a mechanism (called gamer equivalent (GE)) used to simulate the behavior of gamers with different levels of ability (i.e., from newbie to expert). The GE is implemented in the UPF module and supports four strategies:

(I) the first strategy mimics the behavior of a newbie. If the avatar is crashing into a wall then it can turn left or right. It will check both directions and choose the one without obstacles;

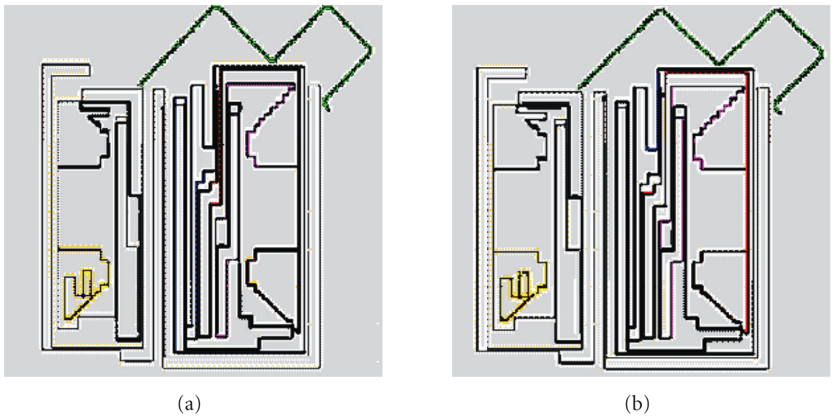

Figure 7: Snapshots of two runs of the same match: (a) unreliable communications, (b) reliable communications.

(II) the second strategy tries to find the direction with the longest free path up to a wall;

(III) the third strategy chooses the direction with the highest number of available paths in the next turn;

(IV) the fourth strategy is very similar to the previous one but it attempts at forecasting the state of next turns instead of just one.

The following performance evaluation is based on 30 trials; in each trial all gamers follow the same strategy. The main results collected are the amount of space occupied by the wakes (generated by motorbikes) and the mean lifetime (see Table 2). For the sake of clearness, in Table 3 only the results obtained by the best strategies are reported. By comparing Tables 2 and 3, we see that the lifetime parameter is more important than the occupied space. For example, strategy (III) is successful versus (I) and (II), even if (I) and (II) have covered a greater percentage of the arena with their wakes. Strategy (III) seems better in exploiting the space near the wakes generated by other gamers.

As described in Section 5, the PS introduces a quite complex timeouts' management that leads to a different timing of actions with respect to a standard game execution. The goal of the last part of the evaluation is to demonstrate that the PS mechanism does not significantly alter the game progress. In this test-bed evaluation, this is done by using in each MM the same gamer equivalent (GE) that has been implemented in the UPF. Mainly, if an action generated by the GE in the UPF does not reach the avatar, then the GE inside the MM will produce exactly the same action. This has been done in order to eliminate any interference due to the MM mechanism. Table 3 shows the results in presence of a reliable and unreliable communication. In the former case, the PL was set to 0%, in the latter, the PL was from 5% to 20%. These results show that, in presence of the PS mechanism, the unreliability of communication does not affect the general outcome of the strategies. The small differences that can be found in Table 3, are due to the random components in the implementation of the GE. To verify this hypothesis, tests have been repeated without the random generators. Figure 7 reports two snapshots of the same match: the first one (left side) is obtained in presence of unreliable communications (20% of PL), the latter (right side) with reliable communications. It is easy to see that the progress of the match and the wakes of the avatars are exactly the same (the very small differences in the visualization are due to the graphic engine). After this result, we can conclude that the performance of a strategy is not altered by the MM, even if the gamer is frequently substituted. Therefore, the PS mechanism has not altered the chances to win of each gamer, and, in this sense, also the C3 condition is verified.

Table 3: Deathmatch of strategies: matches won (%) with reliable and unreliable communications.

Match	Unreliable communications	Reliable communications
(I) versus (III)	0–100	0–100
(II) versus (III)	4–96	0–100
(III) versus (IV)	0–100	8–92

7. CONCLUSIONS AND FUTURE WORK

Virtual environments are an implementation of the metaverse concept, a simulated world populated by a massive number of synthetic avatars that are controlled via Internet. The telecommunication industry is fostering the "ubiquitous participation" of users in virtual environments, producing and marketing powerful mobile devices with wireless capabilities. In this sense, the support of nomadic users in massive metaverses is a very hot topic in research and business. The communication unreliability is one of the main characteristics of wireless technologies and mobile environments. This aspect is very important when dealing with virtual environments, since it can significantly reduce the effectiveness of the distributed architecture and severely limit the playability of the game. In this paper, we propose a new mechanism (called playing session (PS)) that aims at solving this problem, by introducing an architecture capable of dealing with network failures. In order to evaluate our proposal, we implemented a clone of the Amagetron game based on a prototypal implementation of the PS mechanism. The performance evaluation of the case study has shown that the mechanism enhances the playability of the game, while assuring a good level of equity among users.

Future works should extend the prototypal implementation, investigate the subjective expectations of gamers, and consider the cheating problem. From a technical viewpoint, the current implementation of the PS mechanism works in the client-to-server side of the gaming architecture. As a future evolution, the mechanism could be also extended to the server-to-server side: in this case, aiming at reducing the impact of network failures in the communications among servers.

ACKNOWLEDGMENT

Authors would like to thank the anonymous referees for their contributions aimed at improving the paper quality.

REFERENCES

[1] World of Warcraft, 2007, http://www.worldofwarcraft.com/index.xml.

[2] Second Life, Linden Research, Inc., 2005, http://secondlife.com/.

[3] Project Entropia, 2006, http://www.entropiauniverse.com/en/rich/5000.html.

[4] Sociolotron, 2006, http://sociolotron.amerabyte.com/website-2/intro.htm.

[5] S. Cacciaguerra, S. Mirri, M. Roffilli, and P. Salomoni, "Let me participate! Using intelligent agents to support inclusive playing for gamers in disadvantaged conditions," *WSEAS Transactions on Communications*, vol. 5, no. 10, pp. 1973–1980, 2006.

[6] M. Furini, "Mobile games: what to expect in the near future," in *Proceedings of GAMEON Conference on Simulation and AI in Computer Games*, Bologna, Italy, November 2007.

[7] A. Chandler and J. Finney, "On the effects of loose causal consistency in mobile multiplayer games," in *Proceedings of the 4th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 1–11, Hawthorne, NY, USA, October 2005.

[8] J. Brun, F. Safaei, and P. Boustead, "Managing latency and fairness in networked games," *Communications of the ACM*, vol. 49, no. 11, pp. 46–51, 2006.

[9] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The effects of loss and latency on user performance in unreal tournament 2003," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 144–151, Portland, Ore, USA, August 2004.

[10] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in warcraft III," in *Proceedings of the 2nd Workshop on Network and System Support for Games (NetGames '03)*, pp. 3–14, Redwood City, Calif, USA, May 2003.

[11] K. Farkas, D. Budke, B. Plattner, O. Wellnitz, and L. Wolf, "QoS extensions to mobile ad hoc routing supporting real-time applications," in *Proceedings of the IEEE International Conference on Computer Systems and Applications (AICCSA '06)*, pp. 54–61, Dubai, UAE, March 2006.

[12] L. Pantel and L. Wolf, "On the suitability of dead reckoning schemes for games," in *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames '02)*, pp. 79–84, Braunschweig, Germany, April 2002.

[13] S. Cacciaguerra and M. Roffilli, "Agent-based participatory simulation activities for the emergence of complex social behaviours," in *Proceedings of the Social Intelligence and Interaction in Animals, Robots and Agents (AISB '05)*, pp. 1–29, Hatfield, England, April 2005.

[14] B. Mac Namee, *Proactive persistent agents: using situational intelligence to create support characters in character-centric computer games*, Ph.D. dissertation, University of Dublin, Dublin, Ireland, 2004.

[15] D. Livingstone, "Turing's test and believable AI in games," *Computers in Entertainment*, vol. 4, no. 1, article 6, pp. 1–13, 2006.

[16] S. Mc Glinchey and D. Livingstone, "What believability testing can tell us," in *Proceedings of the Conference on Game AI, Design and Education (CGAIDE '04)*, p. 273, Reading, UK, November 2004.

[17] J. Smed and H. Hakonen, "Three concepts for light-weight communication in multiplayer games," in *Proceedings of the 1st International Digital Games Conference (iDiG '06)*, pp. 199–202, Portalegre, Portugal, September 2006.

[18] T. G. Dietterich, "Machine learning research: four current directions," *AI Magazine*, vol. 18, no. 4, pp. 97–136, 1997.

[19] P. Riley, "SPADES: a system for parallel-agent, discrete-event simulation," *AI Magazine*, vol. 24, no. 2, pp. 41–42, 2003.

[20] Armagetron, 2005, http://armagetronad.net/.

[21] S. Cacciaguerra, *Experiences with synthetic network emulation for complex IP based networks*, Ph.D. dissertation, University of Bologna, Bologna, Italy, 2005.