On the Impact of Delay on Real-Time Multiplayer Games

Lothar Pantel
Pantronics
Saarstr. 73
69151 Neckargemünd, Germany
lothar.pantel@pantronics.com

Lars C. Wolf
Technical University Braunschweig
Muehlenpfordtstraße 23
38106 Braunschweig, Germany
wolf@ibr.cs.tu-bs.de

ABSTRACT

Multiplayer games, i.e., games where several persons interact simultaneously over networks like the Internet, receive much interest. One of the reasons is that competing with human counterparts is typically considered as much more interesting and challenging than playing just against a computer.

A major problem of network-based multiplayer games is caused by the network transmission delay. This means that it takes a while until information, e.g., about the movement of the opponents objects and their new position, reaches the receivers. This delay causes several difficulties and leads to paradoxical situations. For example, consider a racing game with two players, shortly after the start both believe that they have the lead because it takes a while until the position of the counter player reaches the local player. Approaches to provide for a global consistent state of the game by introducing a local presentation delay have been proposed, however, these increase the application-level delay even more.

Therefore, it is important to investigate the impact such delays can have on the performance of multiplayer games and the attractiveness of these games for the human players. Such a study is the purpose of this work. We concentrate on real-time games for the Internet where significant delays can occur. The evaluation is performed through measurements using a car racing simulator.

Categories and Subject Descriptors

D.4.4 [**Operating Systems**]: Communications Management – *Buffering*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Artificial, augmented, and virtual realities, Evaluation/methodology*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Benchmarking, Ergonomics, Evaluation/methodology, Haptic I/O*; J.7 [**Computers in Other Systems**]: *Command and control, Consumer products, Real time*; K.8.0 [**Personal Computing**]: General – *Games*

General Terms

Experimentation, Human Factors.

Keywords

Multiplayer games, real-time applications, delay, user impact

NOSSDAV'02, May 12-14, 2002, Miami, Florida, USA.

1 INTRODUCTION

Computer-based games are a big market and there are expectations from various sides that this will increase even more. Multiplayer games where several persons interact simultaneously receive much interest since competing with human counterparts is typically considered as much more interesting and challenging than playing just against a computer [8]. Reasons for this are that humans play often more intelligent, more spontaneous, and with more intuition. If this is not already sufficient, the mere knowledge that there is another human on the other side is challenging.

The first multiplayer games existed already in the 1980 using a split-screen-modus – the screen was split in 2 or 4 windows, one per player. Yet, these systems where centralized on one computer without any distribution. Later systems provided for LAN-based sessions.

A major problem of networked multiplayer games is caused by the network transmission delay. This means that it takes a while until the information, e.g., about the new position of the opponents objects, reaches the receivers. While some technical means such as network-level quality of service mechanisms, e.g., DiffServ, can reduce or at least bound this delay, some delay will always exist, for instance about 0.1s for the propagation of light between Europe and Australia. This delay causes several difficulties and leads to paradoxical situations. As an example, consider a racing game with two players. Shortly after the start both believe that they have the lead because it takes a while until the position of the counter player reaches the local player. Thus, a primary goal is to provide for a global consistent state of the game – independent of the existing delays. Approaches which introduce a local presentation delay, e.g., as that used within MiMaze [1], have been developed to provide for a consistent, distributed state. However, these increase the application-level delay even more.

Since delay cannot be avoided completely, it is important to investigate the impact such delay can have on the performance of multiplayer games. This is the purpose of this work. While we have been working on consistency-enhancing mechanisms [5], these are beyond the scope of this paper and will be presented elsewhere. We concentrate on real-time games for the Internet where significant delays can occur. Especially critical are simulator-like games such as racing games, flight simulators, and sports games because these require very smooth movements of the objects. We neither study non-real-time games nor split-screen mode games or those which are restricted to LANs where typically only very small delays occur.

It should be noted here that not only from a technical point of view, the boundary between games and 'serious' applications is not always a sharp one. For example, flight simulators may be seen as games but also as (semi) professional 'training tools'. Hence, we believe that the results discussed in this work cannot only be applied to games but also to more serious applications.

The outline of this paper is as following: In the next section we discuss multiplayer games and their requirements, the influence of basic system parameters and in particular of delay. In Section 3 we describe our experimental evaluation of network delay handling approaches and the gained results. We discuss related work in Section 4 before concluding the paper in Section 5.

2 MULTIPLAYER GAMES

Multiplayer computer games can follow different general architectures. They can be server based with dynamically determined master or with a central server, e.g., from the game provider. Another approach is to follow a distributed scheme without any central component holding the complete state of the match. The distribution of the information about movements, etc. can be done by multicast communication and every participant's system computes its own view of the global state. It is obvious that the latter approach is better regarding the reduction of transmission delays since all data packets reach the receivers on the direct path.

2.1 Influence of Basic System Parameters

To ensure fairness, identical movements of the game entities (cars, aircrafts, bullets, etc.) must be performed with the same speed on the different endsystems. Obviously, the communication system is a potential source of errors since messages can be delayed or even get lost. The game designer has to decide which messages must be treated in which way. Some are key messages which cannot be computed by other systems. Such messages must, therefore, be transmitted reliably. Other messages are sent regularly such as a new position. If one message is lost, a newer one will replace it soon (and using an absolute position information, errors are avoided). Thus, these messages must not be sent reliably and result in low overhead for sender and receivers. The most critical requirement of distributed games is that of low network delay. Bandwidth needs are relatively low if the number of participants is low. For instance, MiMaze uses a packet size of 52 bytes at a rate of 25Hz.

2.2 Delay

Within this paper, *delay* means the time between the generation of an event and the resulting update of the game state including the presentation to the users. Delay leads to significant problems for real-time multiplayer games. A consistent presentation of the scenario should be given and the distributed state of the game should be similar on all endsystems. Yet, it is an important issue how the system handles old state information and how large the difference between the real and the expected positions of the players in a game may be due to the delay of messages. For instance, in a car racing simulator, position errors occur in dependence of the speed of the cars if no compensation is applied. Hence, resulting questions are: 'How can we, despite the delay, achieve that no car is displayed on an old, wrong position?'; 'How should a collision of vehicles be treated?'.

Many system aspects in real-time multiplayer games are based on assumptions about times. However, a global wall-clock time is not existing in such systems and the forward and backward transmission delays might be different. To reach some common time-base, mechanisms such as NTP (network time protocol) or simple probe-based approaches to determine the skew of clocks can be used.

To get some examples for delay values encountered by existing multiplayer games, we measured the round trip time (RTT) from dial-in home users to a games server. The clients performed delay measurements during their initialization. This way, a player can choose his peer based on the encountered delay. We made RTT measurements to 20 persons during different times of day. Using the largest German Internet service provider as dial-in access point, the average delay was 342ms (with an Intervall [134ms, 660ms]). Via a university dial-in access, we achieved an average of 429ms (interval [154 ms, 745ms]).

2.3 Experiments with Commercial Games

To study the delay-handling approaches used in commercial games, we performed a series of experiments [5] using two popular racing games (Re-Volt and Need-for-Speed) in two player mode. Our goal was to examine how commercial game developers handle the delay-problem, respectively, as to what extent this end-to-end delay is taken into account. The Internet connection used for the experiments had a delay of approximately 200ms. This value is not unusually high, especially for the case that a world-wide utilization should be possible. The test was repeated with a delay of approximately 100ms afterwards – with basically the same results, of course, the observations are somewhat less significant.

The screens of both participants have been placed directly side by side for this test so that deviations could be observed 'globally' at an 'absolute time'.

The following tests were performed in particular:

(a) Start Release

Direct Play (which is used in these games) assigns the role of a coordinator to one of the two participating computers. This way, it can be controlled which computer takes care of tasks which can typically only be provided by one system. This contains also the release of the start (countdown, start-traffic light, start-sound). Since the transmitted start-release signal would arrive delayed (due to network delay) at the computer of the other player, the start on this side would take place later than on the other one if no compensation is applied. A delayed start-release would have obviously massive fairness-problems as consequence. A possible solution is to delay the start-release by an amount equal to the latency between the two computers on the initiating computer as well.

Besides the observation of the start-countdown on both neighboring screens, the start-sound is especially well suited for a check of a correct synchronization. From both loudspeaker systems, the start signal should (within the perceptibility) sound simultaneously.

(b) Start Process

The start is a crucial phase of a race, often making a preliminary decision. As soon as the vehicles begin to move, the first deviations of the position through the speed can occur. Due to the additional acceleration, the deviation is even bigger than with constant speed.

On both computers, two identical vehicles with identically simulated motors, etc. are used to avoid any differences in their accelerations as far as possible. The acceleration is done with the keyboard in order to exclude any different behavior of proportional joysticks. During start, both cars stand at the same line side by side. In principle, it is to be expected that both cars remain head to head also during accelerating (up to the first curve ...). Using this experiment, for the first time it will become clear whether position consistency is remained at all.

(c) Simultaneous Starting (Simulated Start)

To deepen and verify the result from item (b) (Start Process), the two simulated cars are stopped in the course of the race and are aligned at the beginning of a straight route section precisely side by side. Now both are accelerated simultaneously by one person using the two keyboards which are lying side by side in order to guarantee that the keystrokes (within the perceptibility) actually take place simultaneously. Also here the correct behavior would be if the cars would drive side-by-side.

(d) Driving with Constant Speed

In this test, the vehicles drive with approximately constant speed side-by-side. On both systems, an identical situation should be presented, e.g., if one car has a leadership by some car lengths on the

first screen, the same situation should appear on the second screen as well. An identical screen situation only happens if the position consistency is maintained during the game. Without correction, each driver believes to be in front because without a delay compensation, the car of the local player is immediately displayed on the screen while the position of the opponent has become obsolete by t_{delay}. The resulting error $ds = v \cdot t_{delay}$ leads in general to the situation that on both sides the local car is in leadership – a slightly paradoxical situation.

(e) Collision Treatment

A collision as well as a 'crash' of game objects, e.g., vehicles, is often detected in 3D games through a polygon-collision. Only a position difference of 0 of the opposing vehicles (on both screens) leads on both computers to an identical, local evaluation of the polygon-collision. If the position consistency is not sufficient or a delay compensation is missing at all, contradictory situations can occur as illustrated in Figure 1. The deviation (contrary to the driving direction) leads at screen B to a collision, however at screen A this is not the case. Among others, the following results are possible:

- On computer A a collision messages is given, initiated and caused by computer B. Player A is probably confused because on his screen the vehicles are significantly apart.
- Also the following case is possible: On computer B, no collision is generated although both vehicles seem to have collided. The reason for this is that on computer A the vehicles still lie apart.

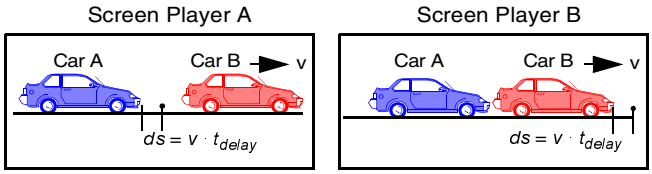

Figure 1: Deviation *ds* results from $v \cdot t_{delay}$ at the opponents car (resp. $a \cdot t_{delay}^2$ considering the acceleration), i.e., car B on screen A and car A on screen B.

With this test (e), it should be examined how the programs respond in the case of a collision. Both vehicles are stopped during the race and placed directly one after another with a short distance. Then on both systems the acceleration is done simultaneously and the vehicles have an identical simulated acceleration performance. The expected result is that both cars accelerate keeping a constant, short distance. If the delay is not compensated, a problem emerges for vehicle B since car A, starting by the delay time too late, is occupying the way. The test is repeated with exchanged roles, vehicle A / B, several times.

Observed Results

The observed results were that the games provided very limited handling of delay effects only. While both provided for a synchronized start release (test 1), already the start process (test 2) was not handled correctly by the first game (due to the game concept, this test was not applicable to the other game): The two identically simulated cars are accelerated simultaneously, nevertheless, the local car was always in leadership, correct would have been that both accelerated cars go directly side by side.

Both games also failed for test 3 and test 4, i.e., each of the two drivers believes to possess the leadership based on the presentation on his screen.

Finally, for test 5, both games were not able to provide any satisfying treatment of collisions. Both showed some handling of delay-induced collisions (e.g., in one case one car is pushed) however on costs of a massive discrimination of the rear car.

3 EXPERIMENTAL EVALUATION OF THE IMPACT OF DELAY

As noted before, an inconsistent state of a game brings several problems and can be confusing to the players. One approach to enhance the consistency is to delay the presentation of the actual game state, e.g., as used in MiMaze and others. However, such methods introduce some further delay, i.e., in addition to the delay already caused by network transmissions and processing. Hence, an important question is how much delay is acceptable for users in real-time games. This user-level delay has to include all communication, processing, and buffering (consistency-enhancing) steps. We performed a series of experiments to study the impact of delay on users, i.e., the performance of players, on the fidelity, and on the attractiveness of the game.

3.1 Principle Setup

The general scenario is given by a network and several computers connected to it, one of them considered as the local machine. The performed steps are shown in Figure 2.

At time t_0, a user-input through the local participant is made, for example from a joystick or a keyboard. This event is evaluated at time t_1 at the local computer and leads here to a position update of the local game object (the local participant). The update of the position may also take place periodically. At time t_2, a timestamp is generated and associated with the object's position which has been determined at time t_1. The timestamp is a globally uniform time or the other systems must be aware of the clock skew between their local times and the clock of the computer generating this timestamp. The data packet consisting of timestamp and object position is sent to all further participants, e.g., via multicast.

The local presentation of the new local object position does not take place immediately. Instead, the presentation is delayed in a queue. This delay is the same and equally long at all participating computers and should be chosen so that it corresponds to the largest appearing delay. However, in the interest of a smooth user control, this delay should be not bigger than necessary, i.e. not bigger than the longest packet transmission time.

While the described steps are performed, data packets containing the actual position information about the objects of the other players are received. They are collected, e.g., by a separate thread, and (based on their timestamps) assigned to the corresponding local position information. Afterwards, the presentation of all the objects belonging to a certain timestamp takes place on the display at time t_4.

With ideal network conditions, i.e., networks with hard transmission time guarantees, this static presentation delay procedure can maintain full position consistency. However, these ideal conditions are given rather seldomly. The disadvantage of this approach is that, in order to perform the synchronization, also the local user input is delayed before the resulting situation is displayed. Hence, the visual reaction to a control input (for example a keystroke) appears only after a downtime. The extent up to which such a delay is acceptable for users will be investigated in this section.

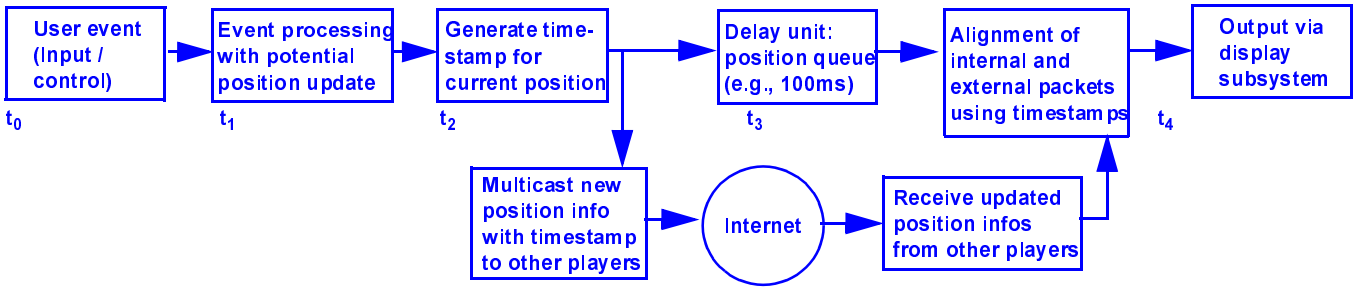

Figure 2: Static Presentation Delay Procedure.

3.2 Implementation

For the evaluation of the described presentation delay scheme, we implemented it in a RC-Car simulation. This game is a virtual racing game for remote controlled model cars. Here, the driver does not use the cockpit perspective but a higher position allowing a view about the whole area (Figure 3). This overview view is especially well suited for investigation purposes since all participants can be seen at the same time. This RC-Car simulator has been developed by one of the authors (L. Pantel).

Figure 3: Virtual RC Racing.

The static delay scheme has been implemented in the 'Virtual RC Racing' game using a FIFO buffer where the delay duration depends on the buffer size. Such a racing game is especially well suited as a test scenario since the closeness to reality depends on the steering behavior. The feeling how the car reacts is influenced by the delay between steering and reaction (in this case the display of the graphics). Overall, we believe that a racing game represents an especially critical case with respect to the time gap between action and reaction.

3.3 Performed Experiments

The purpose of the experiments was to determine the acceptable presentation delay. We had 12 candidates who performed these experiments. In the following, some figures show three lines only, these represent three out of the twelve persons which we considered as representative for the three categories Beginner, Average, and Excellent Driver. Yet, the average values given below include the results from all subjects.

First the candidates performed some training to get accustomed to the behavior of the car and to the game in general. This training was done without any delay. Then the experiments were started. The static presentation delay was increased in steps of 50 ms each up to a maximum of 500 ms. These increments were done steadily, however, the players did not knew about the amount of the artificially introduced delay. The subjects played the game by driving five laps for each delay setup and they were interviewed about their subjective impression of the resulting playing situation. A test series was stopped when the candidate considered the setup as not tolerable anymore, i.e., when the somehow artificial reaction of the car and the behavior of the game in general (caused by the introduced delay) was declared as inacceptable.

The test runs were recorded and examined afterwards using the replay function of the program. This replay offers forward, backward and pause functions and provides us with the necessary means for a thorough analysis of each run. It allows to determine various aspects as discussed in the following:

Average Time per Round in Dependence on Delay: It gets more difficult to control the car if it reacts slower to steering commands (due to the inserted delay). Hence, the driver must drive more slowly and more carefully in order to keep control or he gets into a zigzag course which extends the route since he departs from the best path. In both cases, the average time per round increases. Thus, this value is a good measure for the influence of the presentation delay.

Best Time per Round in Dependence on Delay: The average time also contains exceptions and obvious driving-mistakes, for example through lack of concentration. This value, i.e., the best of five attempts reflects the best value possible for this driver under the given conditions, i.e., with the given presentation delay.

Driven Course: If the speed is not reduced for tests with large delay values, the driver departs more and more from the best course and gets into a zigzag course. This effect also depends on the driving style and cannot be regarded as universal measurement. Moreover, this measure can be grasped only subjectively and was assessed therefore only in 3 steps. The recorded runs were examined using a replay of the driven course and assessed relatively to each other.

Frequency of Leaving the Course: How often the car leaves the overall course depends on the driving style as well. It can be considered as a further increase of the zigzag-course, however, it has the advantage that it can be counted easily and is, hence, a more objective measure.

3.4 Results

Before looking at the subjective impression of the delay impact, more objective and measurable units are given. In Figure 4 and Figure 5, the average and the best lap time per delay value are given for three persons (which have been chosen as typical examples). Figure 6 shows the average over all participants.

Looking at Figure 4 showing the average times, line 1 gives the typical results for an inexperienced person ('beginner'). This line is almost a straight line, but a delay value below 50 ms has no influence on the result. The reason for this line shape is that a beginner tends to drive relatively slowly and, hence, cannot make use of a fast system response time. Therefore, higher delay values up to 200 ms seem to be acceptable for unpracticed drivers.

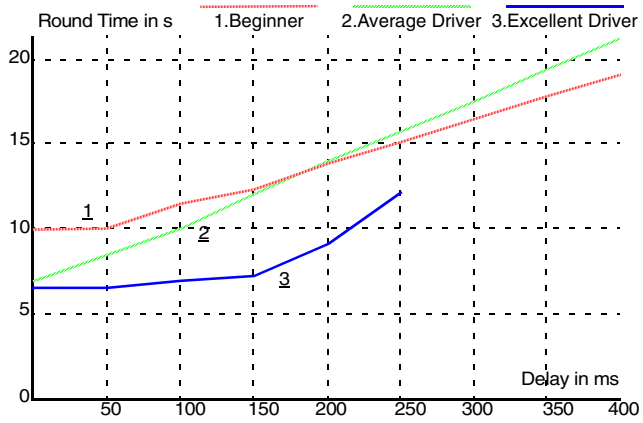

Figure 4: Average Time per Round for Three Typical Test Persons: 1. Beginner, 2. Average Driver, 3. Excellent Driver

The second line represents the results of a medium-level driver with a somehow rough driving style. First, the lap time is quite good since the driver has some training. Then, due to the rough style (with fast reactions to changes in the driving direction of the car), it rises faster than for the beginner. The reason is that this style cannot be successfully continued at higher delay values but leads to a zigzag course. Thus, already in the area of about 50 ms delay the lap times get worse.

The third line shows the lap times of an experienced driver. The difference between best and average lap time is relatively low. Until a delay value of about 150 ms, only a small increase in the round times can be observed, followed by an almost exponential increase above 150 ms. At this point, the reaction time of the car becomes so big that an experienced driver cannot achieve significantly better results than a beginner. The experienced person stops the test series at 250 ms because he feels that the steering does not react reasonable anymore. Below 50 ms, no significant alterations of the lap time are measurable. Hence, we believe that even for competitions, a presentation delay up to 50 ms is uncritical.

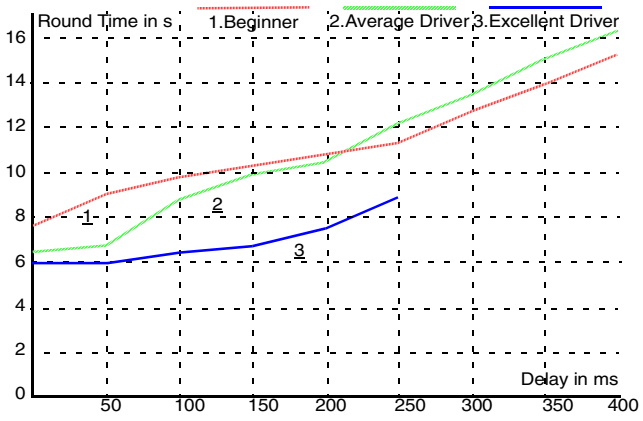

Figure 5: Best Times (of Five Rounds) for Three Typical Test Persons: 1. Beginner, 2. Average Driver, 3. Excellent Driver

Figure 5 presents the results for the same persons but showing the best lap time only. Since no averaging is a applied, a more irregular behavior can be observed. Nevertheless, the results are in line with those shown for the average values in Figure 4.

It should be stated here explicitly that we must distinguish between the objectively measured lap times and the subjective impression of the drivers. Eventually it is not crucial with which (as large as possible) presentation delay still the fastest lap times are possible. Instead it is important how comfortable the drivers feel when they are steering a car as well as whether the game (at a given delay) still looks realistic or at least acceptable.

After the tests, the participants were informed about the achieved results. They were surprised about their relatively good lap times with delay values of 150 ms to 200 ms. Overall it was noticed that it is possible to get used to a presentation delay value until approximately 200 ms – but the driving was stated to be definitely not realistic in such a scenario.

Total-result of the lap times: Figure 6 illustrates the arithmetic mean value of the average-lap time for all participants of the test. Starting for delay values above 50ms, the achieved round times increase strongly. This confirms the previous results that a delay value of 50ms is acceptable without significant restrictions, also, for example, at a competition.

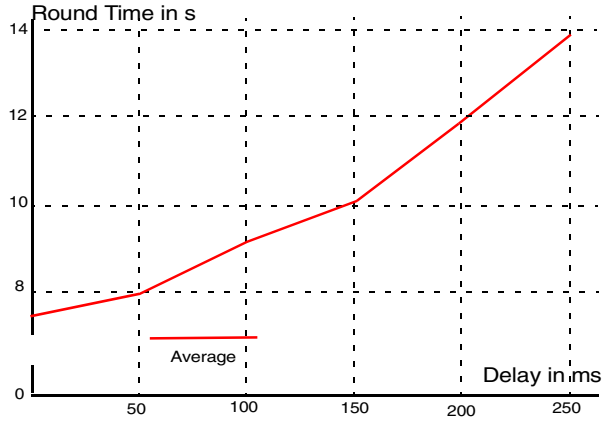

Figure 6: Mean of Average Round Time (Based on Results from all Participants).

Frequency of Leaving the Course per Round: As a further criterion, we evaluated the number of times a car left the course, e.g., because of driving mistakes. The results are illustrated in Figure 7. The y-axis gives the number of times the car run out of the course per lap, hence, it is normally smaller than 1.

In principal, this aspect is a relatively rough and indirect measure and the results also depend on the driving style of a particular person. Nevertheless, Figure 7 shows clearly the steady increase in mistakes made by the drivers. The ability to control the vehicle decreases (somehow linearly) with increasing delay values.

Driven Course: Finally, the driven course has been considered in 3 steps: best, some rolling motion, significant zigzag course. We used the replay function of the application for this evaluation. Of course, this is a subjective measure only. Moreover, the results also depend on the driving behavior of the participants and not only on the applied delay. Thus, no figures are given.

We observed that all those drivers who had some rolling motions already for a delay of approximately 100 ms, entered a zigzag course at latest at a 250 ms delay. Participants which avoided rolling motion at 100 ms through a more cautious, slower style run into such problems not later than at 300 ms.

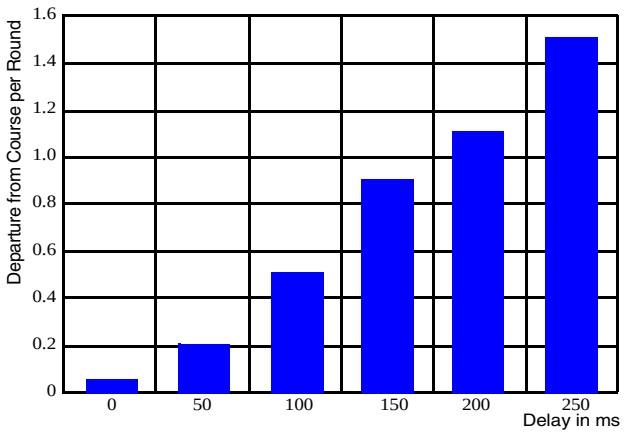

Figure 7: Frequency of Departure from Course.

Subjective Impression: More important than mere objective measurements is the impression the participants had – a car simulator should mainly provide a realistic perception and a game should bring some fun. The statements made by the participants can be summarized as follows (for the overview perspective).

Table 1: Subjective Impression.

Delay	Impression
500 ms	• not acceptable • car cannot be controlled • action and reaction do not fit together
200 ms	• delay is clearly observable • but the car can be controlled • it is possible to adapt the own style and get used to this situation • but the overall behavior is not realistic
100 ms	• acceptable if no high demands with respect to realism are needed • delay can be noticed, but hardly optically be seen
50 ms	• delay can hardly be noticed • the driving behavior is basically unmodified

3.5 View Dependance

The delay value which can be tolerated by the participants depends on the used 'camera' perspective. Using a cockpit view ('sitting in the car') instead of the overview perspective, at least 50 ms of additional delay can be tolerated. The explanation is that the overview perspective places stronger demands on the driver. For instance, the steering direction left/right is flipped when the car goes towards the pilot. Hence, the pilot needs more feedback (action & reaction) from the car. Moreover, in opposite to the cockpit view, in the overview perspective the car can be seen as a whole which also leads to a more sensible perception of the driving behavior.

4 RELATED WORK

As far as we know, not much research work has been performed in the realm of real-time multiplayer games so far. In [3], the problems arising from inconsistent state in participating systems of distributed virtual environments is discussed and a 'timewarp' approach is proposed to handle these. However, no experimental study has been done and no quantitative results are given. The same author presents in [4] a study of consistency in continuous interactive media. Among other parts, an approach called 'local lag' is proposed where the presentation of events is delayed. An experimental study of the effect of these delays is not given since it is not the purpose of that paper.

A similar but also restricted approach is used within MiMaze [1]. MiMaze is a distributed '3D Pacman game' where information about the new state of the game is distributed via multicast to all participants. Based on the received messages, each participant calculates its own view of the global system state. The consistency of the game is improved by delaying the presentation of events using 'buckets'. Basically, this can be seen as a static version of the above mentioned local lag approach. Due to this static defering of the processing in MiMaze, the whole game is always 100 ms in the past. Players which encounter a higher delay, i.e. which cannot be synchronized within this 100 ms range, cannot participate. As an interesting analogy we can notice that this synchronization mechanism resembles the playout buffer used by audio-conferencing systems to reduce jitter effects.

The MiMaze developers performed several measurements of the consistency of this distributed application – which can be seen as related to the usability and performance of the game. The measurements were restricted to participants in the national french network. Especially the position consistency, the difference between the target and the actual position of the game figures, is of major interest. The measured average delay was μ=55.5 ms, the standard deviation is relatively large with σ=50.4. For 65% of the frames there is no deviation of the current position against the target value. This value looks relatively good. However, it is mainly caused by the principal approach of the game where there are long periods of time without any movements (only 10% of the buckets are completely filled with new information). In 85% of the cases, the error is less than 20 units. In comparison to that, the radius of a game figure amounts to 32 units. Yet, there are also considerable exceptions with much more than (several times) the diameter of the game figure.

Unfortunately, more user oriented studies, e.g., how they considered these delay-induced delays, whether that had an influence on their playing, their performance, etc. have not been made.

Recent studies on delay distributions to game servers have been published [2]. However, since these games are not as time sensitive as ours, the results are not directly comparable.

5 CONCLUSIONS

The measurements show that a delay up to 50 ms is uncritical for a car-racing game. This has been shown by the objective measurements as well as by the spontaneous statements of the participants. A delay of more than 100 ms should be avoided, at least for a racing game in the overview view (as studied here) and if the system should provide for some realistic driving behavior. Since we are considering the racing game as a worst-case system, for other games, e.g., first person shooter, such a presentation delay of 100 ms or even more may be acceptable.

These delay values may also be seen in the light of these comparisons: For the lip synchronization requirements of audio and video streams an area of [-80 ms, +80 ms] has been found as tolerable in [7] and if the skew is above 160 ms, basically everybody

detects this error. Moreover for the transmission and processing of signals coming from the different parts of the human eye there exists a delay in the area of about 100 ms and the brain seems to be capable to adapt to such delays (and perhaps another quantum of this size).

In general we believe that static delay schemes are not very well suited for real-time multiplayer games. Schemes using an adaptive mechanism for the presentation delay are a relatively straight-forward extension of this approach. They can improve over static schemes because the current situation and not a, perhaps even not existing, worst-case delay is taken into account. A further improvement is possible by a combination of such an adaptive scheme with dead-reckoning and an extrapolation of the current state to a future state.

As shown in [6], dead-reckoning has the drawback that the prediction error increases significantly with increasing network delays. In that study, also using a RC-car game, the average predicition error was 17cm for a delay of 100ms and 60cm for a delay of 200ms (which is a factor of 3.5!).

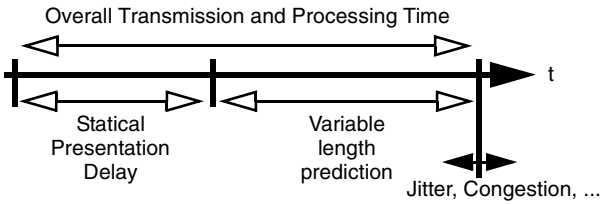

Figure 8: Hybrid Scheme.

Therefore, a hybrid concept using a combination of the two approaches seems to be useful. This can use a shorter prediction interval with a presentation delay of, e.g., 50 ms. Such an approach should also be able to provide for a satisfying treatment of larger delays, e.g., as they can occur in dial-up networks. As illustrated in Figure 8, this hybrid scheme consists of a static presentation delay component (with a length depending on the delay acceptable for the specific game) and a variable length prediciton. The latter has to absorb jitter effects.

A detailed study of the hybrid scheme in general and of the impact of all further schemes (adaptive, dead-reckoning, hybrid) on player performance remains for further study. Other future work items are the treatment of congestion and of collisions within multiplayer games.

References

[1] L. Gautier and C. Diot: “Design and Evaluation of MiMaze, a Multiplayer Game on the Internet”, Proc. IEEE Multimedia (ICMCS’98), Austin, TX, USA, 1998, pp. 233-236.

[2] Tristan Henderson: “Latency and User Behaviour on a Multiplayer Game Server”, Proc.Third International Workshop on Networked Group Communication (NGC2001), November 7-9, 2001, UCL, London, UK.

[3] Martin Mauve: “How to Keep a Dead Man from Shooting”, Proc. of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS) 2000, Enschede, The Netherlands, 2000, pp. 199-204.

[4] Martin Mauve: “Consistency in Replicated Continuous Interactive Media”, Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW) 2000, Philadelphia, PA, USA, 2000, pp. 181-190.

[5] Lothar Pantel: “Approaches for the Treatment of End-to-End Delay within Multiplayer Games” (in German), Studienarbeit, TU Darmstadt, 2000.

[6] Lothar Pantel, Lars Wolf: “On the Suitability of Dead Reckoning Schemes for Games”, First Workshop on Network and System Support for Games (NetGames2002), April 16-17, 2002, Braunschweig, Germany.

[7] R. Steinmetz und C. Engler: “Human Perception of Media Synchronization”, Technical Report 43.9310, IBM European Networking Center Heidelberg, Heidelberg, Germany, 1993.

[8] Jose Pablo Zagal, Miguel Nussbaum, Ricardo Rosas: “A Model to Support the Design of Multiplayer Games”, Presence, Vol. 9, No. 5, October 2000, pp. 448-462.