The Effects of Consistency Maintenance Methods on Player Experience and Performance in Networked Games

Cheryl Savery[1], T. C. Nicholas Graham[1], Carl Gutwin[2] and Michelle Brown[3]

[1]School of Computing
Queen's University
Kingston, ON, Canada
{savery | graham}@cs.queensu.ca

[2]Dept. of Computer Science
University of Saskatchewan
Saskatoon, SK, Canada
gutwin@cs.usask.ca

[3]Dept. of Computer Science
York University
Toronto, ON, Canada
brown@cse.yorku.ca

ABSTRACT

Network lag is a fact of life for networked games. Lag can cause game states to diverge at different nodes in the network, making it difficult to maintain the illusion of a single shared space. Traditional lag compensation techniques help reduce inconsistency in networked games; however, these techniques do not address what to do when states actually have diverged. Traditional consistency maintenance (CM) does not specify how to make game-critical decisions when players' views of the shared state are different, nor does it indicate how to repair inconsistencies. These two issues – decision-making and error repair – can have substantial effects on players' gaming experience. To address this shortcoming, we have characterized a range of algorithmic choices for decision-making and error repair. We report on a study confirming that these algorithms can have significant effects on player experience and performance, and showing that they are often more important than degree of consistency itself.

Author Keywords

Consistency maintenance; game development; usability

ACM Classification Keywords

H.5.3Information Interfaces and Presentation: CSCW.

General Terms

Human Factors; Algorithms

INTRODUCTION

Real-time networked systems, such as multi-player games or shared-workspace groupware, often follow the principle of *distribution transparency* [26] - they attempt to provide an experience that is the same over distance as it is when people are in the same location. The illusion of transparency often breaks down, however, due to problems in the network, such as latency, jitter, or packet loss - often collectively referred to by gamers as 'lag.' In these situations, those parts of the shared environment (e.g., the game world of a multi-player game) that are stored at each node in the network diverge, leading to strange phenomena such as bullets not hitting their targets, objects jumping across space, or avatars moving in a halting and jerky fashion.

Most networked games (and other kinds of distributed systems) combat these problems with lag compensation techniques that try to ensure that information at different nodes in the network (e.g., the locations and status of players in the game world) is in the same state. There are several types of techniques that operate on different principles: for example, some use prediction to try to overcome latency, some use 'local lag' to slow down input to match the latency in the network, and some use time offsets to reduce the variance in remote update times. These techniques work well in many situations - shown both by research studies (e.g., [1,20,25]) and by their adoption by virtually all commercial networked games.

However, the focus of these techniques on 'simple consistency' does not address all of the issues caused by lag and diverging state. In the case of networked games, we have identified two additional factors that are important to player experience and performance, but that are not adequately considered by most consistency maintenance schemes. First, lag compensation techniques are never perfect, and so games need to make decisions in the presence of inconsistency. Second, when the techniques recognize a state divergence and attempt to correct it, the error must be repaired at one (or more) of the nodes.

The first issue is called the *decision-making* problem. When the game must make a fast decision about a game-critical event (e.g., whether a bullet hits a target), it must often do so with the assumption that states at different nodes have diverged. Therefore, the game must choose one of the different states to use as the basis for the decision. This choice could mean substantially different outcomes for game events, which could have a large effect on understandability and player experience.

The second issue is called the *error repair* problem. When a state divergence is identified (e.g., when a prediction algorithm has moved an avatar to the wrong location), the error must be repaired – and since errors often involve a

CSCW'14, February 15–19, 2014, Baltimore, Maryland, USA.

http://dx.doi.org/10.1145/2531602.2531616

visible object in the game, the repair must be visually understandable. Some techniques provide smoother corrections (e.g., interpolating between the incorrect and correct locations over a time period) but at the cost of reduced consistency, in that it takes longer for the two nodes to reach the same state.

Understanding the effects of the decision-making problem and the error repair problem are important in the design of a game's overall approach to consistency maintenance, but there is little information available about how important these issues are, and how they affect player performance and experience. To address this shortfall, we carried out an experiment to address three questions:

- Is simple consistency the most important issue for player experience in networked games?
- What are the effects of different choices about what state is used to make game-critical decisions?
- What are the effects of different choices about how errors are corrected?

Our study tested 26 people (13 experts and 13 novices) in three different custom games that implemented several different techniques for lag compensation, decision-making, and error repair. We recorded both player performance and subjective assessments of play experience. Our study confirms that decision-making and error repair are important design considerations for networked games. Players preferred conditions where game-critical decisions were made such that outcomes were more understandable to the local player. Players also preferred smooth animation over jerky movements caused by instantaneous 'warps' when the corrections were small. However, when larger corrections were required, players only marginally preferred smooth corrections compared to warping. These results show that simple consistency is not the most important element in player experience, since both smooth corrections and local-player understandability actually reduce the overall state consistency of the game.

Our study also provides several additional results that add to our knowledge of player experience in conditions of lag. For example, we found that there were large differences in the degree to which experts and novices noticed the effects of the consistency techniques; we also found that some players enjoyed the challenge of having to adjust for the lag, particularly when playing against a weaker opponent.

Our work makes three important contributions to the understanding and design of CM schemes for networked games. First, we demonstrate that state consistency alone does not address all of the issues that arise from lag. Second, we present an integrated CM design space that brings lag compensation together with the additional techniques of decision-making and error repair. Third, we provide the first empirical evidence of how these factors affect player experience and performance in three different game genres, and identify trade-offs in the techniques' use.

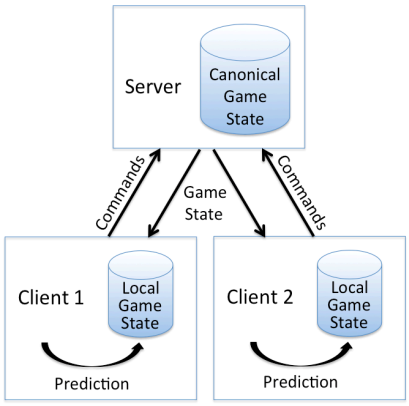

Figure 1. Typical client-server architecture: server maintains canonical game state; local clients use prediction and server updates to determine local game state.

BACKGROUND AND RELATED WORK

In this section, we describe the consistency maintenance problem and briefly review existing research related to consistency maintenance in networked games.

The Consistency Maintenance Problem

In multiplayer games, inconsistencies arise due to a combination of network latency and the use of prediction by the client to determine the local game state. Because of the real-time performance requirements of many forms of networked games, some inconsistency must typically be tolerated in exchange for faster feedback to the player and quicker resolution of game events.

Most networked games are based on an authoritative central server (Figure 1). To combat cheating, game clients are generally not allowed to make any decisions that affect player performance in the game [15]. Player inputs are passed to the server, which determines the new game state and propagates it back to the clients. Due to network latency, the client cannot wait for the next game state to arrive from the server before providing feedback to the player. Thus, the client must use prediction to calculate the local game state. Problems occur when the client's predicted state differs from the server state. For example, consider a racing game where two players attempt to move their car to the same position at the same time. Each player sees that the position is open, and locally the client predicts that the move can occur. However, when the server receives the commands, it determines that a collision has occurred. The server sends the corrected game state to the clients, which are then forced to repair this inconsistency. The use of multiple servers can reduce the latency between the server and client, but introduces the complexity of maintaining a consistent state among the servers [7].

Inconsistencies can have a negative impact on player performance and experience [4,19], such as the confusion caused by sudden warps in position, the frustration of shooting directly at another player and missing [5], or the perplexity of suffering damage from a grenade while protected by an invincibility shield [2].

Lag and Consistency in Games

Research on consistency in games can be divided into two categories: studies that look at the general effects of latency on player performance and experience [4,10,11,19], and research into specific CM techniques such as dead reckoning [1,21] and local lag [9,25]. These techniques are described in detail in the Reducing Inconsistency section.

Claypool [11] generalizes the effects of latency in networked games by categorizing game actions along two dimensions: precision (the accuracy required in performing an action) and deadline (the time period within which an action must be performed). This provides us with an understanding of the types of game actions that are most affected by network latency, but provides no guidance on techniques that can mitigate the effects of latency.

The Human Factors framework [22] discusses three aspects of multiplayer games that must be considered when determining CM requirements. First, there are the types of entities found in the shared environment. E.g., objects such as avatars typically have tighter consistency requirements than other objects in the game world. Second are the types of interactions players can have with those entities, particularly whether interaction with multiple players is possible and whether the interaction affects game critical variables. The framework then considers how player states may diverge in time and space and in rate of change. This work provides a good foundation for making CM choices, but focuses primarily on the single dimension of the CM problem of reducing inconsistency.

AN EXPANDED CONSISTENCY DESIGN SPACE

Unlike the work described above, we believe that there are three factors to be considered in the design of a CM scheme: in addition to the traditional factor of reducing inconsistency, we highlight the importance of decision-making and error repair. These factors are related and it is important to deal with them as a whole rather than looking at each factor individually. We introduce three axes (Figure 2), describing a range of approaches that help with each problem, and concluding with a discussion of how decisions in each axis influence the behavior of the others.

Reducing Inconsistency

The first line of defense in the CM problem is to deploy an algorithm that prevents or reduces inconsistency in the first place. While a core set of algorithms has been proposed, it is not yet fully understood in what situations different algorithms should be applied. In general, these algorithms trade off game responsiveness, the frequency with which the algorithm causes jarring corrections to the player's view, and the degree to which the views of different players diverge [22]. These algorithms follow three broad strategies, all of which have been used in commercial games: using *prediction* to anticipate remote updates; *delaying input* to allow simultaneous execution on all players' computers, and *offsetting time* to allow state to diverge in a controlled fashion.

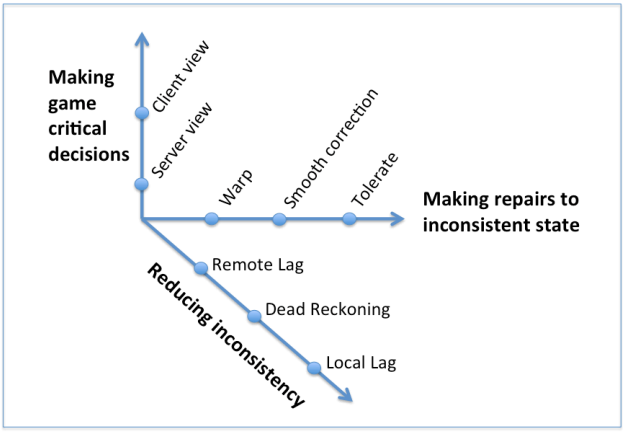

Figure 2. The design space consists of three axes capturing the issues that must be addressed in deciding on a consistency maintenance scheme

More specifically:

- *Predictive techniques,* such as dead reckoning [16,20], predict changes to remote state before those changes are propagated over the network. The predicted state can be used locally, compensating for the time to transmit the state over a network. However, if the prediction is incorrect, the local state must be updated, possibly in a jarring or confusing manner. Dead reckoning is commonly found in role-playing games such as EverQuest and World of Warcraft. Here, game clients predict the position of remote entities based on their last known position and velocity.

- *Delayed input techniques,* such as bucket synchronization [6] and local lag [17], delay local actions to allow simultaneous execution by all clients. For example, if a player presses the "W" key to move forward, he will not see his avatar move until after the delay period, providing time for the input to be sent over the network to remote game clients. Delayed input algorithms improve consistency, at the cost of slower response to player actions.
- *Time-offsetting techniques,* such as remote lag [5], apply a constant delay to updates from remote clients. For example, in the Half-Life first person shooter game, remote players' avatars are lagged by a constant 100ms. Time-offsetting increases inconsistency (since remote state updates are not applied immediately), but makes the degree of consistency predictable, something to which players may be able to adapt.

Each of these strategies embodies trade-offs. Prediction can increase consistency when the prediction is correct, at the cost of potentially jarring corrections when the prediction is wrong. Delayed input increases consistency at the cost of reduced responsiveness. Time-offsetting lends predictability to inconsistency, but increases inconsistency overall. Because time-offsetting displays an exact replica of the remote state which has been simply offset in time, its use can reduce or even eliminate the need for error repairs.

A growing body of research attempts to determine the conditions under which the various techniques are applicable [9,21,25]. For example, local lag is effective up to about 100ms of lag [25,9]; bucket synchronization is most effective in real-time strategy games where users can tolerate small delays [11]. However, little work has been done on the criteria for selecting one algorithm over another.

Decision-Making

Game-critical events are highly visible occurrences with significant impact on the game's progression. Examples include picking up a piece of treasure, crashing into an opponent's car and blowing up, or defeating a "boss" enemy to complete a level. Game-critical events are different from, for example, moving an avatar, which can generally be easily undone or corrected, and where small differences in positions may be unnoticeable.

The resolution of game-critical events in the presence of inconsistency can have an enormous impact on player experience and performance. In a shooter game, one player might be frustrated to miss an enemy who was clearly in his cross-hairs, while another might be upset at being hit while hiding behind a rock. When inconsistency causes players to see the world differently, the outcome of such game critical events can have unintuitive and negative consequences.

An important question in resolving game-critical events is the choice of perspective to use. If game clients all have a different state, which state should be used to resolve the outcome? One important case involves actions which have a source and a target: e.g., one player shoots another, or throws a pass to another. The game-critical questions to be resolved are whether the target player was shot, or whether the target player caught the ball. Three choices of perspective are available:

- *Server perspective.* The game server maintains a canonical state. The event is resolved relative to this state, even if it disagrees with what the players see.
- *Source client perspective:* The decision is based on what the initiator of the action sees. For example, Counter Strike uses this approach for shooting, in the belief that it is more noticeable to the shooter if he erroneously misses, than to the target if he is erroneously hit [5].
- *Target client perspective* resolves events based on what the target of the action sees. This is used, for example, in Halo: Reach for determining damage when a player has activated Armor Lock [2], an ability providing the player with a few seconds of invincibility. During this period, a player would be frustrated if a grenade thrown toward her inflicted damage.

Note that the choice of perspective does not necessarily imply the node within a distributed system where choices are made. For example, Counter Strike uses the shooter perspective, but the hit decision is actually made on the server (and therefore requires the server to be able to reconstruct what the shooter saw when firing) [5].

The perspective used for resolving hit decisions can have an enormous impact on player experience. Dick et al. have shown that in Unreal Tournament 2004, network latency above 100ms has a significant impact on game score, while in Counter Strike, player skill was the primary determinant of game score [12]. A primary difference between these games is that Unreal Tournament 2004 uses server-perspective for resolving hit decisions, while Counter Strike uses the perspective of the shooter.

When choosing a strategy for decision-making, game developers should consider which player is more likely to notice an unintuitive result, and which player is more negatively impacted by an erroneous determination.

Error Repair

The final axis of the design space deals with how to repair inconsistencies when they are discovered. Three basic approaches are used to resolve inconsistencies:

- *Correct immediately:* The simplest solution is to immediately update the local state to the newly arrived correct state. This may result in jarring visual artifacts, such as remote players "warping" to a new location. This warping can be confusing, and can distract the player's attention to the location of the abrupt change.
- *Correct smoothly:* Here, the incorrect local state is progressively updated to the correct state. For example, a remote avatar might quickly run to the correct location rather than simply warping. This removes the effect of jarring corrections at the cost of prolonging the period of inconsistency.
- *Tolerate:* If the inconsistency doesn't matter, it may be preferable to leave it as-is. For example, the positions of players in Blizzard's World of Warcraft are often inconsistent, with little effect on gameplay.

There are significant tradeoffs between these solutions, and to-date, these have received little attention. Correcting immediately may affect players' immersion in the game world since a player's attention will be drawn toward any sudden discontinuities in the position or motion of objects in the game [18]. A player may also lose context, for example, if an avatar that was in front of him is suddenly behind him and now out of his field of view.

Smooth corrections may reduce the jarring effect of corrections [24]; however, they prolong the inconsistency, which may be unacceptable in some game situations. Fiedler suggests the rule of thumb of moving 10% toward the true position during each frame, but if the correction is large, simply warping to the new position [13]. More sophisticated forms of smooth correction are possible; e.g. varying priorities may be given to different objects in the game to ensure the objects of higher importance deviate the least from their true position [8]. To date, there has been little effort to characterize when smooth corrections are effective, and how they should be applied.

Design Consequences

Designers need to be aware of how decisions made in one axis can affect the range of choices in other axes. We now list some examples of possible interactions. Our study, presented in the next section, explicitly tests the effects of these interactions on players' experience and performance.

- The use of local lag has positive effects on the decision-making and error repair axes. Local lag reduces inconsistency, therefore reducing the magnitude of corrections and increasing the likelihood that client and server perspectives are the same when decisions are made. Local lag may be a good choice in situations where corrections are particularly bad, e.g., to avoid race conditions when two players are concurrently picking up the same item.
- Prediction can lead to large state divergences. This can increase the importance of the choice of perspective used for decision-making, and can lead to an increasing frequency and magnitude of corrections. Prediction may be most appropriate in situations where an accurate prediction algorithm is available and where corrections do not negatively impact gameplay.
- The suitability of the remote lag algorithm depends upon the nature of game-critical decisions. If one player is more impacted by the outcome of a game decision, as in our shooter example, remote lag can allow the decision to be made from that client's perspective. However, if all players care equally about a decision, having divergent views of the game world can be confusing. For example, in a two-person racing game, if each player saw a delayed version of the other player, both players might believe that they were the first over the finish line.
- Smooth corrections may reduce the jarring effect of warps, but increase state divergence versus immediate correction of inconsistencies when they are discovered. Smooth corrections may therefore be a poor choice when state consistency is very important.

USER STUDY: TYING THE DESIGN SPACE TO USER EXPERIENCE AND PERFORMANCE

We carried out a study to provide a better understanding of how the dimensions of the design space affect player experience and performance. In particular, we designed the study to answer four questions:

Q1. Does better overall consistency in a distributed game always lead to better experience?
Q2. How does the decision perspective (client or server) affect player perception of critical game events?
Q3. How do different error repair strategies (warping or smooth correction) change experience?
Q4. Does the player's level of experience affect the results of Q1-Q3?

We answered these questions with three custom networked games designed specifically for this study. The games replicated critical aspects of real multiplayer game situations (but were otherwise kept simple to avoid the effects of strategy). The three games were *Paddle Blasters*, a cooperative canoeing game, *Eliminate*, a first-person shooter, and *Speed Daemons*, a racing game.

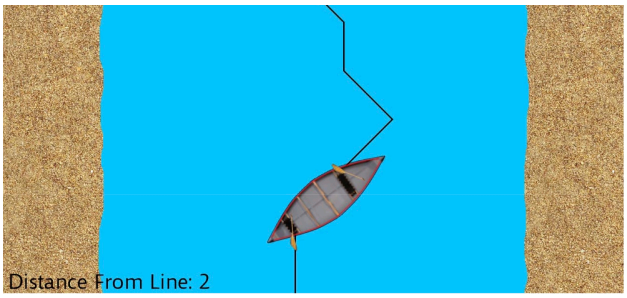

Figure 3. Paddle Blasters: Players cooperate to paddle a canoe and follow a path indicated by a black line.

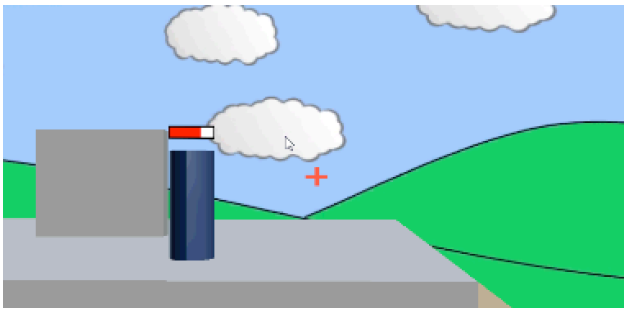

Figure 4. Eliminate: FPS in which players attempt to hit their opponent while avoiding being hit themselves.

Paddle Blasters is a cooperative two-player game that was used to investigate whether overall consistency led to better player experience (Q1). In Paddle Blasters, players attempt to paddle a canoe down a river; the goal is to keep the canoe as close as possible to the black line that zigzags down the river (Figure 3). Each player paddles on one side of the canoe. If one player is paddling, the canoe turns; when both players paddle, the canoe moves ahead in its current direction.

Eliminate is a first-person shooter game used to investigate the effect of using different perspectives for game-critical decisions (Q2). In the game, players attempt to shoot their opponent while avoiding incoming shots. Simple block avatars are located on a platform separated from the other player by an open area that cannot be crossed (Figure 4). This restricts the movement of each player and forces the player to focus on the shooting task. To discourage firing without aiming, players must wait three seconds between shots.

Speed Daemons is a 2-D side scrolling game used to study two techniques for repairing location errors: immediate warping and smooth correction (Q3). Each player controls a racecar and attempts to pick up coins while avoiding mud puddles (Figure 5). Players earn one point for each coin they collect; colliding with puddles causes a loss of two points. The game scrolls from right to left at a constant rate with new coins and mud puddles appearing from the right.

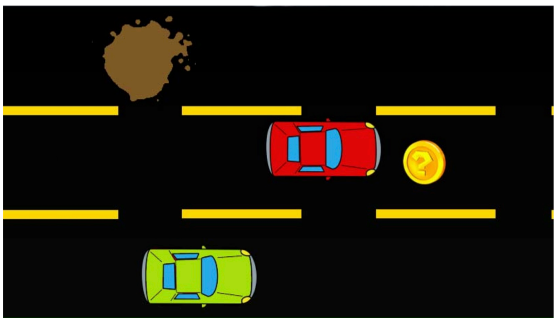

Figure 5. Speed Daemons: Racing game in which players attempt to collect coins while avoiding mud puddles.

Study Methods

Participants

26 participants were recruited from a local university. 13 of the participants were frequent and experienced gamers, and 13 were non-gamers. We classed participants as gamers if they played fast-paced games at least once per week. We did not require expertise in any given game. Although expertise might be specific to an individual game or game genre, we hypothesized that there is significant commonality between games in, for example, the ability to rapidly manipulate game controls, or to visually process game events. In addition, because our games are designed to focus on common game elements (moving, targeting, avoiding), there is a reasonable expectation that expertise will apply across all of our scenarios. We use gaming experience as a factor in the study, as described below.

Procedure

Participants played in pairs, seated at separate computers and able to communicate through a headset. For each game scenario, the players were shown the game and allowed to practice for up to five minutes. During this learning period no latency was added to the game. Typical network latency during the practice sessions, as measured by ping time, was 1ms or lower.

Following the training period, in order to simulate typical wide area network conditions, synthetic latency was introduced on the network. Based on work by Armitage [3] showing that players of games such as Quake 3 actively select servers where the latency is less than 150 to 180ms, we chose to limit the synthetic latency to 200ms. The latency was randomly distributed between 50 and 200ms according to a Poisson distribution. Because the games used synthetically created latency over a local area network, there were no large latency spikes during the trials. The participants played several trials. Each trial used a different CM technique, corresponding to the conditions from questions 1-3 above. The order of the trials was randomized and the conditions were balanced for order. Specifics of the trial conditions for each game scenario are described below.

Paddle Blasters (Q1). Participants played three different conditions. Each condition was repeated three times for one minute each. In condition A, 200ms of remote lag was used (*Lowest Consistency* between the two players' views); in condition B, 100ms of local lag was used in addition to dead-reckoning-based prediction (*Medium Consistency*); and in condition C, 200ms of local lag was used (*Highest Consistency*). In condition B, warping was applied to repair any incorrect states that occurred when latency exceeded 100ms. No information regarding the specific differences between conditions was provided to the participants.

Eliminate (Q2). There were two five-minute trials, both of which used 200ms of remote lag for displaying the position of the remote avatar. In trial A, hits were determined based on the shooter's perspective (*Shooter*), while in trial B, hits were determined based on the canonical game state as determined by an authoritative central server (*Server*). Expert gamers typically play their game of choice for many hours each week, and over weeks, they become intimately familiar with the algorithms being used in the game. As each condition was played for only five minutes, we told the participants what scheme was being used in order to simulate the experience they would have in extended play in real life.

Speed Daemons (Q3). Participants played three one-minute trials. In both trials A and B, dead reckoning was used to predict the position of the remote car. However, in trial A, immediate position warping (*Warping*) was used to fix incorrect positions, and in trial B, smooth interpolation-based correction techniques were applied (*Smooth*). In trial C, remote lag of 200ms was applied to the remote car position, meaning that no corrections were required for the position of the remote car. Smooth corrections were used to repair any incorrect states for the local car caused by collisions between the two cars (*Local Only*). No information regarding the specific differences between conditions was provided to the participants.

After each of the trials in each of the games, participants answered several Likert-style questions about their experience in that particular trial condition (Tables 1, 2, and 3). After all the trials for each game, participants were also asked to indicate which of the conditions they preferred.

Setup and Apparatus

The study used custom software written in C#, XNA, and the Janus toolkit [23], and ran on two computers connected via a dedicated network. The system maintained a frame rate of approximately 60 FPS. Each game used two clients and a server. The server arbitrated conflicts between the two clients (e.g., if both players picked up a coin at the same time in Speed Daemons).

Study Design

We analysed each game separately, using mixed-factorial ANOVAs to test for effects on performance measures. One within-participants factor was determined for each question (i.e., *Degree of Consistency* for Q1, *Decision Perspective* for Q2, and *Correction Method* for Q3, and *Expertise* (Gamer or Non-Gamer) was used as a between-participants

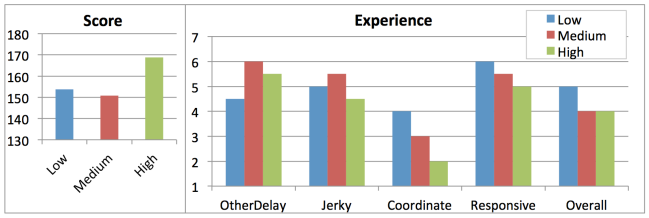

Figure 6. Paddle Blasters: Overall performance (means) (low score is best) and questionnaire measures (medians), by degree of consistency.

OtherDelay: The other player's paddling seemed delayed.
Jerky: The canoe moved in an unexpected jerky manner.
Coordinate: It was easy to coordinate my paddling with the other player.
Responsive: The canoe movement was responsive when I pressed the space bar.
Overall: Overall this version was as good as the practice version.

Table 1. Player Experience questions for Paddle Blasters

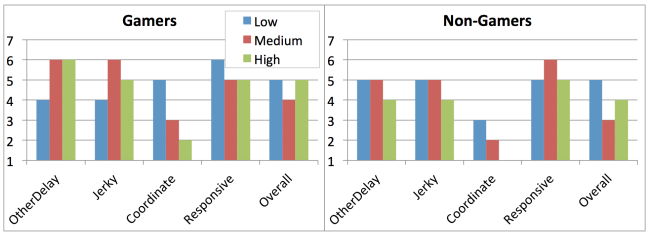

Figure 7. Paddle Blasters: Questionnaire median responses by expertise (Gamer / Non-Gamer) and degree of consistency

factor. Player experience (the questions of Tables 1, 2, and 3) was analysed with Wilcoxon tests on medians.

In the following sections, we present our results. For each question, we first consider the performance measures, and then the player experience questions.

Results

***Q1**: Does better consistency lead to better **performance**?*

The Paddle Blasters game compared three different consistency maintenance schemes, providing different amounts of overall consistency between the two clients' views. Remote Lag produces the least overall consistency because the remote player's actions are delayed by 200ms. Local Lag produces the highest consistency because both players' actions are delayed by the same amount, so both players have an identical view of the game. The hybrid approach provides consistency between these two extremes.

We analysed the effects of *Degree of Consistency* on the pair's shared score (we used the mean of the pair's best two scores for each technique; note that lower scores are better in this game). One-way within-subjects ANOVA showed a significant effect of *Degree of Consistency* ($F_{(2,49)}$=3.81, p=0.038) (see Figure 6). Follow-up pairwise t-tests indicated that scores were significantly worse in the *High Consistency* condition (200ms Local Lag) compared to the *Medium* and *Low Consistency* conditions (p<0.05).

***Q1**: Does better consistency lead to better **experience**?*

For the experience questions (Figure 6), Friedman tests showed a main effect of *Degree of Consistency* for the 'overall' question (χ^2=6.72, p=0.035) and near-significant effects for the 'jerkiness' question (χ^2=5.51, p=0.063) and the 'responsiveness' question (χ^2=5.51, p=0.063) (Fig. 6).

In addition, we divided the data by *Expertise* into Gamer and Non-Gamer groups (Figure 7). For Non-Gamers, Friedman tests show no effect of *Degree of Consistency* on any question. For Gamers, Friedman tests showed a main effect of *Degree of Consistency* for the 'jerkiness' question (χ^2=6.78, p=0.034) and a near-significant effect for the 'overall' question (χ^2=5.90, p=0.052). Follow up Wilcoxon tests showed that for the jerkiness question, the *Medium Consistency* condition was significantly worse than either *Low* or *High Consistency* (p<0.05). For the 'overall' question, Wilcoxon tests showed no differences.

Following the trials, the participants indicated their overall preference among the three techniques. For Non-Gamers, five chose the *Low Consistency* condition (Remote Lag), two chose *Medium Consistency* (Local Lag plus prediction), and six chose *High Consistency* (Local Lag). For Gamers, seven chose *Low Consistency*, two chose *Medium Consistency*, and four chose *High Consistency*.

***Q1** – Interpretation of Results*

In Paddle Blasters, the *High Consistency* condition (Local Lag) resulted in the worst overall score. This indicates that overall consistency does not necessarily result in the best performance. Interestingly, however, the results were not as clear-cut for player experience, with 10 players selecting the *High Consistency* condition as the best overall and 12 selecting the *Low Consistency* condition.

Contrary to previous evidence that 100ms is the limit for local lag for direct interaction [25], we found that fewer than half of the participants noticed 200ms of local lag. Answers to the 'responsiveness' question (Table 1) showed that 15 of the 26 participants found the *High Consistency* (Local Lag) condition to be as or more responsive than the *Low Consistency* (RemoteLag) condition. We believe that the game type has an effect on the noticeability of local lag. The fact that the local player was not the only one controlling the canoe meant that most players (even the gamers) did not notice the 200ms of local delay. However, even though the participants did not always notice the lag, it affected their score in the game.

The *Medium Consistency* condition resulted in the best overall score. However, only four of the 26 participants selected this as their preferred version of the game. This counter-intuitive result is best explained by one participant's comment: "The jerkiness of [this condition] was off-putting, but it played the best."

Our results show that players have definite opinions about the play experience of different CM techniques, but that these are not strongly related to degree of consistency (players preferred either the least or the most consistent version). In addition, it is clear that the visual jerkiness of

the game is an important factor (an issue we consider in more detail below).

Q2**: Effects of decision perspective on **performance

The Eliminate game was used to explore the importance of the perspective from which game-critical decisions were made. The two conditions were *Server*, where decisions are made based on the server view of the game, and *Shooter*, in which decisions are made based on what the shooter saw. The *Server* condition requires players to lead or predict the position of a moving enemy in order to score a hit. We analysed the effects of factor *Decision Perspective* on game performance (individual accuracy and score).

A 2x2 ANOVA showed a main effect of *Decision Perspective* on score ($F_{1,24}$ =4.39, p=0.042) and on accuracy ($F_{1,24}$ =8.00, p=0.007). As shown in Figure 8, both score and accuracy were lower when decisions were made based on the *Server* viewpoint.

ANOVA did not show a main effect of *Expertise* on either score ($F_{1,24}$=2.83, p=0.10) or accuracy ($F_{1,24}$=.282, p=0.60). Differences between Gamers and Non-Gamers are shown in Figure 8. There was no interaction between *Decision Perspective* and *Expertise* on score ($F_{1,24}$ =0.68, p=0.41) or accuracy ($F_{1,24}$ =0.165, p=0.69).

Q2**: Effects of decision perspective on **experience

For the experience questions (Figure 9), Wilcoxon tests showed a main effect of *Decision Perspective* (Z=-2.42, p=0.015) only for the question "Aiming at the other player was easy." Since Wilcoxon tests do not determine interactions, we divided the data by *Expertise*, and carried out secondary analyses with the Gamer and Non-Gamer data (Figure 10).

For Non-Gamers, Wilcoxon tests did not show a main effect of *Decision Perspective* for any of the questions. However, for the Gamers, Wilcoxon tests showed a main effect of *Decision Perspective* for three questions: "Aiming at the other player was easy." (Z=-2.62, p=0.009), "The game felt delayed or laggy." (Z=-2.02, p=0.044) and "There were no issues with delay in this game" (Z=-2.26, p=0.024). In all three cases, participant responses favoured the *Shooter* condition.

Following the game trials, participants were asked which condition they preferred. Among Non-Gamers, 8 preferred the *Shooter* condition and 5 preferred the *Server* condition; for Gamers, 10 preferred *Shooter* and 3 preferred *Server*.

Interview comments show that for some players, the added challenge of leading when aiming (in the *Server* condition) increased enjoyment as opposed to causing frustration. For example, player 13 commented, "I preferred Game A [*Shooter*] but compensating for the lag in Game B was actually kind of fun"; player 1 stated, "Game B is better because of the increased potential for improvement of skills and increased challenge."

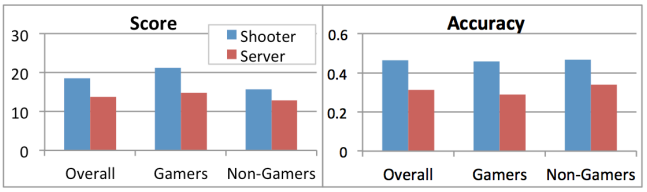

Figure 8. Eliminate: Mean performance by expertise and decision perspective (Server / Shooter).

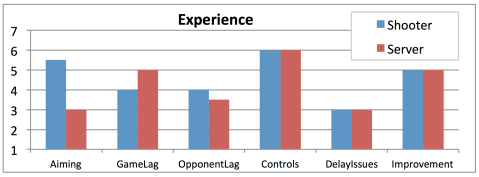

Figure 9. Eliminate: Questionnaire median responses by decision perspective (Server/Shooter)

Aiming: Aiming at the other player was easy
GameLag: The game felt delayed or laggy.
OpponentLag: The other player seemed delayed
Controls: The mouse and keyboard controls worked well.
DelayIssues: There were no issues with delay in this game.
Improvement: I was getting better at shooting/aiming throughout the game.

Table 2. Player Experience questions for Eliminate

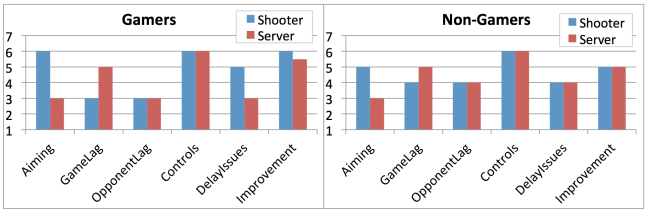

Figure 10. Questionnaire median responses by expertise (Gamer / Non-Gamer) and by decision perspective

***Q2** – Interpretation of Results*

The choice of decision perspective had a significant impact on player performance, with score and accuracy being significantly better in the *Shooter* condition. This is not surprising, as in this condition, the player scores a hit whenever he shoots with his cross-hairs over the enemy's avatar.

This performance difference was seen in both Gamers and Non-Gamers. Surprisingly, score and accuracy measures for the Gamers were not significantly better than for the Non-Gamers. We attribute this to the fact that the Gamers were frequently paired with other Gamers who were also better at avoiding being hit. When we analyzed separately the five trials in which Gamers were paired with Non-Gamers, we did find a significant effect of *Expertise* on both score ($F_{1,8}$ =13.98, p=0.002) and accuracy ($F_{1,8}$ =14.77, p=0.001) with Gamers performing significantly better than the Non-Gamers.

Gamers were more aware of the effects of latency. For the 'game lag' question, they reported that the *Server* condition felt significantly more laggy and for the 'delay issues' question they reported that the *Shooter* condition had fewer

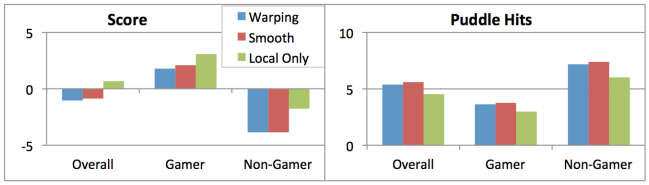

Figure 11. Speed Daemons: Performance (means) by expertise (Gamer / Non-Gamer) and by correction method

OtherJump: It was annoying when the other car jumped or moved unexpectedly.
MyJump: It was annoying when my car jumped or moved unexpectedly.
DelayIssues: There were no issues with delay in this game
Overall: Overall this version was as good as the practice version.

Table 3. Player Experience questions for Speed Daemons

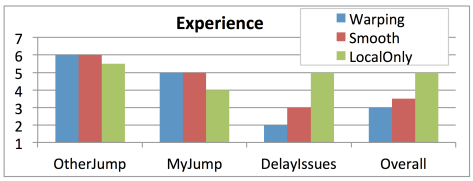

Figure 12. Speed Daemons: Questionnaire median responses by correction method

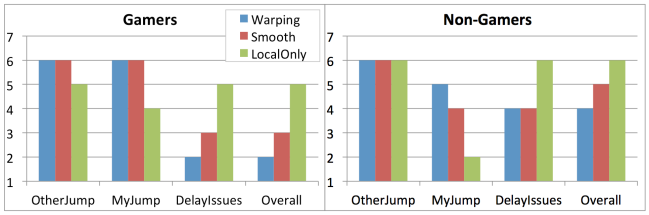

Figure 13. Speed Daemons: Questionnaire median responses by expertise (Gamer / Non-Gamer) and by correction method

issues with delay. Non-Gamers responses for these questions showed no significant difference between the conditions. However, despite not subjectively seeing a difference, Non-Gamers' performance was affected by the choice of technique. This suggests that Non-Gamers may not notice subtle differences in CM techniques because they are too busy with general gameplay, but that these differences may nevertheless impact their performance.

A main lesson is that consistency between players' views may be less important than perceived local correctness (i.e., intuitive resolution of game-critical events).

Q3**: Effects of error repair strategies on **performance

The Speed Daemons game was used to examine the difference between three error repair techniques: instantaneous correction (*Warping*), interpolated correction (*Smooth*), and a technique that avoided all corrections of the remote car (*Local Only*). We analysed the effects of *Correction Method* on individual performance (game score and puddle hits).

ANOVA did not show a main effect of *Correction Method* on either player score ($F_{2,72}$=0.20, p=0.82) or puddle hits ($F_{2.72}$=0.29, p=0.75) and there was no interaction between *Correction Method* and *Expertise* on either measure (score: $F_{2,72}$=0.02, p=0.98; puddle hits: $F_{2.72}$=0.02, p=0.98).

ANOVA did show a main effect of *Expertise* on both score ($F_{1,72}$=5.12,p=0.027) and puddle hits ($F_{1,72}$=7.62, p=0.007). As shown in Figure 11, the score was higher for Gamers and the number of puddle hits was lower.

Q3**: Effects of error repair strategies on **experience

For the experience questions (Figure 12), Friedman tests showed a main effect of *Correction Method* for the last three of the four questions (see Table 3). Since Friedman tests do not determine interactions, we divided the data by *Expertise*, and carried out secondary analyses with the Gamer and Non-Gamer data (Figure 13).

For the Gamers, Friedman tests showed a main effect of *Correction Method* for all of the last three questions: for the 'other jump' question, (χ^2=7.37, p=0.025); for the 'delay issues' question, (χ^2=10.3, p=0.006); and for the 'overall' question, (χ^2=13.0, p=0.002). Follow-up Wilcoxon tests showed that for the 'delay issues' question, *Warping* was significantly worse than both *Smooth* and *Local Only* and that *Smooth* was significantly worse than *Local Only* (p<0.05). For the 'overall' question, both *Warping* and *Smooth* were significantly worse than *Local Only* (p<0.05).

For Non-Gamers, Friedman tests only showed a main effect of *Correction Method* (χ^2=8.19, p=0.017), for the third question "There were no issues with delay in this game". Follow up Wilcoxon tests showed that both *Warping* and *Smooth* were worse than *Local Only* (p<0.05).

Following the trials, the participants indicated their overall preferences. Among Non-Gamers, participants were evenly split (each technique was chosen by four people, with one participant stating no preference). Among Gamers, three people preferred *Warping*, two preferred *Smooth*, seven preferred *Local Only*, and one did not state a preference.

***Q3** – Interpretation of Results*

In Speed Daemons, we saw that player performance was not affected by the choice of technique. We attribute this to two aspects of the game. First, collecting the coins was a very simple task and during each trial all the coins were always collected. Second, the mud puddles were large and players could find themselves cornered into a position where the puddle was unavoidable. Getting into such a position was affected more by player skill than by the choice of technique.

Surprisingly, smooth corrections provided only a marginally better player experience than warping. (Gamers found there to be more issues with delay in warping than smooth corrections, but otherwise there were no significant differences between the two.) We attribute this to the large magnitude of corrections in Speed Daemons and to the very visible nature of the corrections. Corrections occurred when the cars collided with each other and the local client predicted that they did not, or vice versa. When a correction occurred, it was possible for the cars to be on the wrong sides of each other. To do the correction smoothly, the cars quickly drove around each other. Participants' lukewarm

response to smooth corrections suggests that these large, rapid movements may be as problematic as instant warping.

In the *Local Only* condition (which used Remote Lag), the position of the remote car was always correct, just delayed in time. The local client can still make incorrect predictions about whether the local car collided with the remote car, and thus there may still be corrections to the local car when the true remote position arrives over the network. However, these corrections were smaller and less frequent than with dead reckoning, and had less impact on player experience.

As with Eliminate, gamers and non-gamers had similar experiences with the different techniques, although gamers continued to be more sensitive to the differences.

The clear winner for player experience is Remote Lag, which reduced the number and magnitude of corrections. Remote Lag leads to greater divergence in state among the participants, but generally leads to smoother animation. This is another indication that factors other than overall consistency can be critical to player experience.

SUMMARY OF RESULTS

To summarize, the following are the important lessons from the study:

- The algorithm that leads to the best player performance does not always lead to the best player experience. In both Paddle Blasters and Eliminate, we saw a decoupling of player performance and experience. In Paddle Blasters, medium consistency was chosen by only 4 of 26 players as their preferred algorithm, yet delivered the best score (in a statistical tie with low consistency). While most players in Eliminate preferred the direct aiming that allowed them to score more hits, some preferred the leading version that provided more challenge.
- Similarly, the algorithm providing the best consistency does not always lead to the best player performance or the best player experience. In Paddle Blaster, the high consistency condition led to lowest scores and worst experience.
- In all three games, we saw that expert gamers are more likely to perceive anomalous behavior due to latency than are novice players. Surprisingly, novice players can fail to distinguish negative behavior even when their performance suffers from it. This was most evident in Eliminate where the novice players had significantly worse accuracy in aiming in the *Server* condition, but reported no significant difference when responding to the experience questions.
- Smooth corrections are not always better than warping, particularly for large corrections that may be required after collisions between entities. In Paddle Blasters, the participants reacted negatively to the small warps in position that caused jerky motion. However, in Speed Daemons, smooth corrections were found to be only marginally better than warping. The large corrections where a car moved quickly around the other car were disconcerting, even when they were done smoothly.

We next discuss the conclusions we can draw from these findings.

DISCUSSION

Our study shows that the CM problem is multi-dimensional, as captured in the three dimensions of our design space. The three example games show that focusing on consistency alone is insufficient, and that the factors of decision-making and error repair are important in designing a complete CM scheme. The study indicates that designers must consider questions such as the following:

- How important is consistency to decision-making? For example, in a scenario where two players attempt to pick up the same object at the same time, it is critical that all clients make the same decision, and a high-consistency CM scheme should be adopted, such as local lag.
- To what degree can decision-making be viewed as asymmetric, in the sense that the results of the decision are more critical to one player than another? If this is the case, then the remote lag CM scheme can be used.
- To what degree can corrections be tolerated? For example, in games where player movement is highly predictable, dead reckoning schemes may be appropriate as they will result in few corrections, and these can be easily and (often) imperceptibly repaired. However, if the corrections are highly noticeable, remote lag and local lag, both of which reduce the number and size of corrections, may be more suitable.

The multi-dimensional nature of the CM problem highlights the importance of user testing to determine the true effects of algorithmic choices on player experience. As we have seen, the best player experience is not always provided by the algorithm that leads to the highest consistency, or even the best player scores. We saw two reasons why the participants actually preferred games where they scored worse. First, part of the fun of playing a game is the challenge of trying to accomplish a goal. If the game is too easy, players may find it boring and not as fun as a more challenging version. For this reason, we believe some of the gamers enjoyed the version of Eliminate where they could not aim directly at their opponent, but instead had to determine how much to lead them. Second, in one condition of Paddle Blasters, warps were used to correct the canoe position as quickly as possible which allowed the players to score better than in the other two conditions. However, the warps lead to jerky animation that many players found to be visually annoying.

Play testing also showed the fallacy of assuming that smooth corrections would always provide a superior experience over warping. Testing showed that remote lag was a surprisingly good option in many situations, as it negated the need for most of the corrections.

The study shows that the type of game and the target player group are important considerations in developing a CM strategy. Games designed for casual players may be able to

use a simpler strategy, as many of the subtle effects may go unnoticed by the players. However, fast paced games designed for expert gamers must carefully consider the impact of CM choices on each interaction.

We have seen that there are interesting interactions between the three axes. The use of remote lag can reduce the number of corrections making it easier to provide a smooth animation. However, remote lag also leads to the greatest state divergence and can make it more difficult to make game critical decisions that appear intuitive to all players.

Dead reckoning has the potential to help with this and works well when the movements of the game entities are highly predictable. However, when an entity moves erratically, a large number of jarring corrections can result. Smooth correction techniques can mask the negative effects of the error repairs, but when the game uses physics and entities collide with each other, the corrections may be too large to mask with smooth corrections.

In a perfect world, game developers would prefer complete consistency among all players' views of the game world. However, as a consequence of the tight performance requirements of most networked games, we must accept a degree of inconsistency and sometimes even make incorrect decisions. Ultimately, the key to a good consistency maintenance strategy is providing players with an intuitive local view of the game world. Similar to the findings of Greenberg and Marwood [14] for some classes of real time groupware, it is not necessary that each player have an identical view of the state, merely that the views are consistent enough and that the outcomes of game critical decisions can be rationalized by the players.

CONCLUSIONS AND FUTURE WORK

Our research has provided an expanded view of CM in networked games, focusing on two factors – decision-making and error repair – that can help designers improve player experience. We have performed a user study confirming that it is not sufficient to focus on consistency alone. While it is difficult to generalize the results of game performance studies because changing even small parameters in a game can possibly result in quite different results, we claim that it is productive to analyze individual games in terms of the three axes of the design space, and that surprising results can be obtained from these analyses. From our study, we saw that consistency algorithms providing the best consistency do not necessarily lead to the best user experience. We found that players preferred smooth animation and preferred the results of game-critical decisions to be consistent with their view of the game, even when this resulted in lower overall consistency. Other surprising results included that smooth corrections were not always successful in masking error corrections, and that novice players often failed to noticed the effects of lag even when it caused their performance to suffer.

In future work, we will carry out additional studies using the framework, exploring these axes in real-world gaming situations, and investigating techniques that can better manage the tradeoffs identified in our study. We plan to further investigate interaction effects among the axes such as the relationship between remote lag and corrections and whether the perspective for game critical decisions affects the choice of correction technique.

ACKNOWLEDGEMENTS

We gratefully acknowledge the funding of the GRAND Network of Centres of Excellence.

REFERENCES

1. Aggarwal, S., Banavar, H., and Khandelwal, A. Accuracy in dead-reckoning based distributed multi-player games. *SIGCOMM Workshops*, (2004), 161–165.
2. Aldridge, D. I shot you first! Gameplay networking in Halo: Reach. *Game Developers Conference*, (2011).
3. Armitage, G., An experimental estimation of latency sensitivity in multiplayer Quake 3. Networks, (2003), 137–141
4. Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J., Agu, E., and Claypool, M. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. *SIGCOMM'04 Workshops*, (2003), 144–151.
5. Bernier, Y.W. Latency compensation methods in client/server in-game protocol design and optimization. *Game Developers Conference*, (2001).
6. Bettner, P. and Terrano, M. 1500 archers on a 28.8 network: Programming in the Age of Empires and beyond. *Game Developers Conference*, (2001).
7. Brun, J., Safaei, F., and Boustead, P., Managing latency and fairness in networked games. Commun. ACM 49, 11 (2006), 46-51.
8. Chandler, A. and Finney, J. On the effects of loose causal consistency in mobile multiplayer games. *NetGames '05*, (2005), 1–11.
9. Chen, H., Chen, L., and Chen, G. Effects of local-lag mechanism on task performance in a desktop CVE system. *JCST, Vol. 20*, 3 (2005), 396–401.
10. Claypool, M. The effect of latency on user performance in real-time strategy games. Elsevier Computer Networks, Vol 49, (2005) 52–70.
11. Claypool, M. and Claypool, K. Latency and player actions in online games. *CACM 49*, 11 (2006), 40–45.
12. Dick, M., Wellnitz, O., and Wolf, L. Analysis of factors affecting players' performance and perception in multiplayer games. *NetGames'05*, (2005), 1–7.
13. Fiedler, G. Networked Physics. 2006. gafferongames.com/game-physics/networked-physics/.
14. Greenberg, S. and Marwood, D. Real time groupware as a distributed system: Concurrency control and its effect on the interface. *CSCW*, (1994), 207–217.
15. Hogland, G. and McGraw, G. *Exploiting online games: Cheating massively distributed systems*. Addison-Wesley, 2007.

16. IEEE. IEEE Standard for Distributed Interactive Simulation - Application Protocols. *IEEE Standard 1278.1-1995 (revision of IEEE Std 1278-1993)*, 1995.
17. Mauve, M., Vogel, J., Hilt, V., and Effelsberg, W. Local-lag and timewarp : Providing consistency for replicated continuous applications. *Transactions on Multimedia 6*, 1 (2004), 47–57.
18. Murphy, C. Believable dead reckoning for networked games. In E. Lengyel, ed., *Game Engine Gems, Volume 2*. CRC Press, 2011, 307–328.
19. Nichols, J. and Claypool, M. The effects of latency on online Madden NFL football. *NOSSDAV '04*, 146–151.
20. Pantel, L. and Wolf, L.C. On the suitability of dead reckoning schemes for games. *NetGames'02*, (2002), 79–84.
21. Pantel, L. and Wolf, L.C. On the impact of delay on real-time multiplayer games. *Proc NOSSDAV* (2002), 23–29.
22. Savery, C., Graham, T.C.N., and Gutwin, C. The human factors of consistency maintenance in multiplayer computer games. *GROUP'10*, (2010), 187–196.
23. Savery, C. and Graham, T.C.N. Timelines: Simplifying the programming of lag compensation for the next generation of networked games. *Multimedia Systems*, (2012).
24. Smed, J. and Hakonen, H. *Algorithms and networking for computer games*. Wiley, 2006.
25. Stuckel, D. and Gutwin, C. The effects of local lag on tightly-coupled interaction in distributed groupware. *CSCW'08*, (2008), 447–456.
26. Zhao, S., Li, D., Lu, T., and Gu, N. Back to the future: A hybrid approach to transparent sharing of video games over the internet in real time. *CSCW'11*, (2011), 187–196.