On the Effects of Loose Causal Consistency in Mobile Multiplayer Games

Angie Chandler, Joe Finney
Computing Department
Infolab 21, South Drive
Lancaster University, UK
Tel: +44 1524 510325
{angie, joe}@comp.lancs.ac.uk

ABSTRACT

It is well understood that distributed multiplayer games, as soft real-time systems, require a degree of support from the underlying network in order to function correctly, in terms of predictable end to end bandwidth, latency and jitter. In a mobile environment, such applications face even greater challenges, as the latency of wireless networks is much higher than their wireline counterparts, jitter is often much higher due to network handoff and bandwidth is at a premium. In fact, the latency of many wide area wireless networks is beyond the tolerance of most multiplayer games, rendering such applications unusable.

This paper presents the design and experimental evaluation of Rendezvous, a novel decentralized consistency management mechanism that enables the collaboration of multiple players in mobile real-time games, even in a high latency environment. The operation of the mechanism is validated through the analysis of a real world example - a distributed mobile multiplayer soccer game called Knockabout, which is designed to operate on the Smartphone platform. Experimental results are included not only comparing Rendezvous to an existing consistency mechanism, but also measuring the length of network delay tolerated by the platform and its effect on the players.

Keywords

mobile, wireless, real-time, consistency management, gaming, multiplayer, collaborative, latency, peer to peer, relevance filtering.

1. INTRODUCTION

In recent years, mobile networked gaming platforms have expanded from relative obscurity to becoming truly ubiquitous, with emerging devices such as the Sony PSP and Nintendo DS complimenting the existing Nokia N-Gage and Smart phone platforms. These devices now integrate wireless network technologies including Bluetooth, 802.11, GSM and GPRS into high performance multimedia terminals to enable distributed multiplayer gaming between handsets. These devices pave the way to the next logical step in mobile gaming – *distributed real-time multiplayer mobile games*. However, we are unlikely to see any significant deployment of such games in the near future despite the advanced specifications of these emerging devices. This is largely due to the QoS characteristics of the underlying network technologies, in particular the high latency and low bandwidth associated with the wide area wireless networks needed to make these devices truly ubiquitous.

The usability of real-time multiplayer games is primarily bounded by the latency of the underlying network - devices need to wait for consensus to be reached between themselves on game decisions (i.e. a consistent view of the game to be determined) before they can be visualized on any device (i.e state is committed). The period of time this takes is obviously dominated by the end to end network latency. Moreover, the well documented human tolerance of delay in interactive systems is approximately 200ms [1]. As the latency of many wide area wireless networks is already beyond this limit, it can render existing consistency mechanisms inapplicable to the mobile gaming domain.

This paper proposes the design and evaluation of a novel peer-to-peer consistency mechanism, called Rendezvous, which enables the collaboration of

multiple players in mobile, real-time games in a high latency environment, and provides detailed experimental comparisons between this work and an existing consistency mechanism, going beyond work discussed in our short paper [2]. Unlike existing consistency mechanisms that provide a high degree of consistency in low latency environments, Rendezvous provides the means to tolerate a degree of *managed inconsistency* between game players' views of the game in the short term, thus enabling real-time behaviour. In the longer term, Rendezvous aims to ensure high level consistency between views is reached without penalizing players in terms of fairness or overall game playability. The paper goes on to provide detailed design and experimental evaluation of the Rendezvous mechanism based on prototype laboratory implementation and its application to a distributed soccer game called Knockabout. Results presented include a quantitative analysis of the length of network latency tolerated by the Knockabout game whilst using Rendezvous, and comparisons to the existing consistency mechanism Rollback. Further analysis also considers the relationship between consistency and playability at high latency.

2. THE NEED FOR A NEW CONSISTENCY MODEL

It is readily apparent that in any conceivable circumstance it is necessary to maintain a consistent shared view of the game state between players in order to play a real-time multiplayer game. This requirement applies as much to multiplayer games played on a single shared console as it does to distributed games, wired or wireless. However, we believe the inherently low bandwidth and high latency of today's wide area wireless networks requires a different approach to be taken to maintaining consistency of this shared state.

The most widely used, and highly successful, class of consistency mechanism for distributed real-time multiplayer games is rollback [3][4], the earliest example of which being Timewarp [5][6]. Rollback mechanisms rely on a local lag between the player performing an action within the game and the game actually displaying the consequences of that action. During this local lag, state can be exchanged between the different instances of the game, and a consistent view (or at least compatible views which maintain causal ordering) can be reached. Should any inconsistencies between views be detected, history can be 'rolled back', and rewritten unbeknownst to the user, provided that the events leading to the inconsistency occurred within the period of time covered by the local lag. By this means, even events which initially arrive late or out of sequence can be executed in the correct order with no detriment to the player. However, this mechanism relies exclusively on the arrival of network events within the local lag window, which is necessarily bounded by the human tolerance for delay. Once the network delay is beyond this critical value, *rollback techniques can no longer be applied*. If they are applied, either the local lag is beyond the human tolerance, in which case the real-time nature of the game is removed, or rollback occurs after events have been visualized by the game (in which case the rules of cause and effect are broken, and the player observes arbitrary and visual discontinuities in the game).

A number of studies have been taken of the maximum tolerable latency in gaming for the user, with figures ranging from 120ms to 250ms [1][7], and exact figures depending on the type of game. These figures are critical, because they highlight that existing consistency mechanisms as they stand are inappropriate to the task of maintaining consistency in high latency environments, simply by the nature of their construction. Trials show that the end to end latency of today's wide area wireless networks are far beyond these levels of delay, with the expected round trip time of GPRS networks in the region of 1000-2700ms [8], and whilst its 3G replacements perform better, latencies are still of the order of 400-500ms [9]. This implies that today, and for some time to come, *existing techniques for consistency management of multiplayer games cannot be effectively applied to the real-time mobile gaming domain*.

3. RENDEZVOUS

Rendezvous is a novel, highly optimistic consistency mechanism targeted at the resolution of the current stalemate between the need for consistency and real-time behaviour in high latency collaborative systems, including mobile multiplayer games. The novelty of Rendezvous lies in its ability to tolerate a degree of *managed inconsistency* between views of shared state within a game, rather than attempting to maintain complete and absolute consistency at all times. This tolerance for inconsistency enables Rendezvous to allow each node within the game to visualise actions

as they happen, irrespective of the lag between that device and the rest of the network, with consistency maintained through a shared state convergence mechanism. In other words, Rendezvous permits short term inconsistencies to arise between views to enable real-time behaviour, but controls the flow of the game to enable long term consistency.

3.1 Overview of Consistency Management in Rendezvous

Rendezvous is a support platform which acts in a peer to peer fashion. Each node engaging in a game session executes an instance of the game application, and each of those game instances operates in *complete asynchrony* with the others. Any user input is handled within the local instance of the game. e.g. the decision to move a local player and its visualization operates immediately and in real-time without any external input. The degree of consistency between game instances is maintained through an entity known as the *Rendezvous arbitrator* which is integrated into the game application at compile time. It is the arbitrator that is responsible for converging the views between the local game state and that of remote instances of the game. This is achieved via the application frequently and periodically (typically every frame) passing a serialized version of its game state to the arbitrator in the form of name/value pairs (i.e. "positionX, 345"), with one arbitrator operating independently at each peer and periodically exchanging *target states* with other arbitrators, as illustrated below.

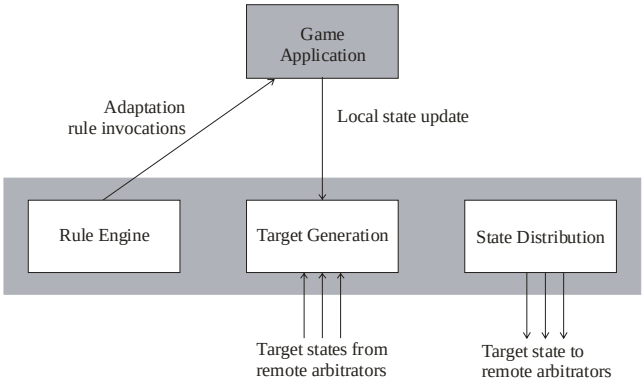

Figure 1 - The Rendezvous Arbitrator

A target state is an optimal point at which all the current (probably differing) states held by the various game instances could once more become consistent. Each arbitrator generates a target state based upon its local game state and the last known target state received from each of the other arbitrators in the session (see Figure 1), and in turn distributes this state to the other arbitrators. It is worth noting that arbitrators never redistribute the local game state – only target states. In essence, arbitrators conspire to generate a consistent shared view not of the current state, but of some *future state* which is compatible with all of the views.

Once a target state has been generated, each node calculates a means to step towards that target state through use of a series of application defined *adaptation rules*, which allow Rendezvous to control non-player entities in the game, and even to subtly bend the rules of the game universe, thus providing each player with a subtly different, but *causally connected*, view of the game. One can consider existing rollback mechanisms as optimized derivatives of lock step synchronization schemes, or transactions. The Rendezvous mechanism lends its roots more to control theory – it views the differing states as a complex system to be balanced according to the set of adaptation rules.

3.2 Generating the Target State

Theoretically, a target state should be calculated based on its reachability from every current state within the game using the available adaptation rules. However, this calculation, in particular when applied to increasing numbers of players, is known to be of high complexity. Furthermore, it is likely that during the execution of a game, the target calculation will be performed a number of times, thus adding an ever larger processing burden on a potentially low power processor, such as those found in mobile devices. Ordinarily, this reachability calculation would remain unavoidable, as it would be necessary to guarantee that all nodes were able to converge on the target state, but the very nature of the Rendezvous concept implies that *no target state will ever be reached*. This is because of the inevitability of the discovery of a further inconsistency prior to final convergence on the target state; the nodes will tend towards the target state, but they are unlikely to ever reach it before a further inconsistency is exposed and a new target is calculated, incorporating the old one.

For the purposes of target calculation, the knowledge that nodes are unlikely to reach the final target state provides an opportunity to simplify the calculations necessary to produce that state into a process of arithmetically combining the individual state values

(such as player location, etc), without the need to consider application semantics at this stage. This process often takes the form mean or median averaging of values, but is also configurable by the application to allow for the handling of less tolerant state (e.g. the game score).

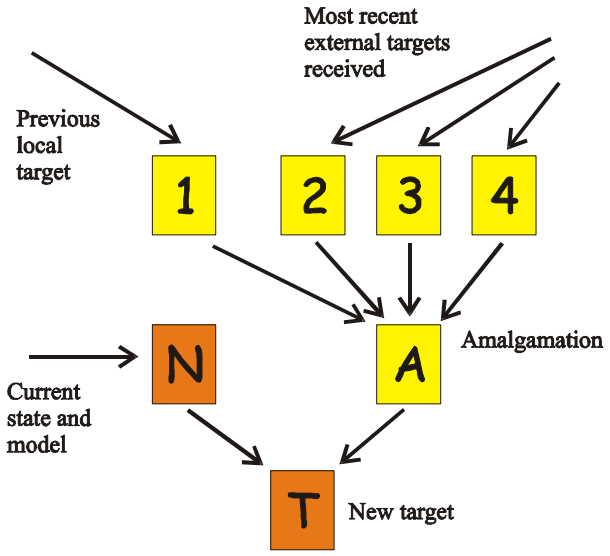

Figure 2 - Calculation of the Target State

Each new target state is calculated from the amalgamation of all previous target states received from all collaborating nodes in the game, calculated as described. This amalgamation is then further combined with the node's current game state, with a given weighting, to generate the target. Once established, the target state is then distributed to all other nodes, where it will be used as an element of each other node's amalgamation during the next round of target state generation. Thus, for instance, the most recent target state generated by player 1 will be sent to each of players 2, 3 and 4 (and also stored locally). This target state will then be used by each of the four players, including player 1, to determine a subsequent target state, with the process continually updating and incorporating new information from each of the players gradually. These changes are also gradually put into effect within the game through the use of the adaptation rules.

3.3 Adaptation Rules

With the generic functionality of the Rendezvous mechanism handled within the arbitrator, the adaptation rules provide necessary application specific methods, without which the target states generated by the arbitrator would be meaningless. The function of the adaptation rules is to provide a low level understanding of the game, with high level decisions remaining strictly in the domain of the player, via the arbitrator.

Adaptation rules are specified by the game application and implemented by the games programmer according to a defined API. They consist of a *precondition range*, specifying a set of conditions under which the rule can be fired, a *postcondition range*, specifying the potential outcomes of the rule, and an *implementation*. These rules are used to model the functionality of a game, and expose the implementation of that functionality to the arbitrator. From here, the arbitrator can choose to invoke the implementation of one or more rules to 'encourage' the local state to converge with the current target state. It is important to note that the arbitrator can only manipulate the game in ways defined (i.e. condoned) by the application programmer, therefore guaranteeing that no sudden changes or discontinuities will occur in any given view.

Which adaptation rules fire is, naturally, determined by the arbitrator from the current state of the game when compared to the target state. This calculation, as previously discussed with respect to generation of a target state, is also theoretically a complex reachability or AI problem, with each rule to be fired needing to prove that the ultimate goal of the target state will remain reachable once its firing has occurred, in order to ensure that local minima in the reachability tree are avoided. However, with the continual update of the target state itself, the difficulties presented by local minima diminish, with any problems occurring likely to rapidly disappear independently of the process of stepping through rules. As a result, more complex calculations are justifiably ignored in favour of the more lightweight and practical firing of the most appropriate rule or rules at each given update.

The choice of rules to be fired is determined not only through matching of projected changes in state with relevant pre and post-conditions, but also through a hierarchy of consistency requirements. For example, in the simple soccer game presented here (see section 4.1), the consistency of the overall score has a significantly higher priority than the accurate positioning of a single player. This priority system can lead to temporarily increased inconsistencies in low priority components of the game, but enables the maintenance of key elements critical to the

consistency of the game as a whole. As with all elements of the adaptation rules, the priorities are specified by the application programmer, but are catered for through the implementation of the arbitrator.

4. IMPLEMENTATION

Evaluation of the Rendezvous mechanism is non trivial, as the effectiveness of the mechanism cannot easily be quantified. Moreover, a simulation of the mechanism would fail to reflect the key aspects of the system. For instance, player actions in a game cannot be easily modelled using existing schemes such as random walk, as they are driven by events in the game, and the impact of how the game players respond to the managed inconsistency could not be accurately evaluated through simulation alone.

In order to evaluate the Rendezvous concepts, we opted to develop a prototype of the mechanism and evaluate it with a real world mobile multiplayer game. The arbitrator was implemented in the Microsoft .NET framework using the C# language, for easy testing development in the PC environments whilst enabling portability to the Smartphone platform to allow us to perform user studies. The prototype implementation currently consists of approximately 7400 lines of code, and has a static memory footprint of around 60 Kbytes. Furthermore, Rendezvous can only be truly analyzed through its effectiveness to an application. In this respect, we have developed a multiplayer game, also designed for the Smartphone platform, called Knockabout. Through this application, we measure the performance of Rendezvous, and analyze its effectiveness.

4.1 Knockabout

Knockabout is a mobile multiplayer soccer game, designed to operate with up to 20 players split into two teams, each interacting with the game via their own gaming handset. Each player in the game controls an individual and unique soccer player in Knockabout. The game provides a simple top down view of the playing area (as shown in Figure 3), and allows players to move around the game area (pitch) and to dribble and kick the ball, with the aim of their team scoring more goals than the opposing team. The game is targeted at the SmartPhone and so the screen size is limited to 176x220 pixels, but with scrolling the overall pitch size is 616x1000. The game was developed in C++ using the Gapidraw graphics libraries.

Knockabout runs asynchronously to the arbitrator, taking input from the user via the keypad and updating the local player instantly. Once per frame (30 times per second), the game state is serialized and delivered to the arbitrator. This state includes the position and trajectory of all players, the ball, and the game score.

Figure 3 – The Knockabout Mobile Multiplayer Game

The game exports adaptation rules and methods to the arbitrator which are in keeping with the rules of the game. This permits the arbitrator to manipulate the actions of any characters in the game other than the one controlled by the local human player in a manner consistent with the gameplay. Adaptation rules were implemented to allow such characters to *run* around the pitch (with no preconditions) and *dribble* and *kick* the ball (with the precondition that the character is close to the ball). A further adaptation rule was added to enable a character to shoot at the goal, with the capacity to override all other targets (such as positioning of characters or the ball), whenever a goal deficit is detected. This overriding goal shooting method exhibits a higher priority than any other adaptation rule, as suggested in the previous section, with any positioning of either the ball or players according to targets becoming secondary until the score is once more consistent. In turn, the positioning of the ball takes higher priority than the positioning of players, with one or more of the players potentially moving further from their target positions in order to kick the ball and return it to its own target position. Together, these prioritized adaptation rules and methods allow the arbitrator to control the local view of the game, with the aim of converging it to the current target state.

Finally, for the purposes of evaluation, Knockabout was modified to incorporate a traditional Rollback based consistency mechanism, including the instigation of a local lag, in order to provide a direct comparison to Rendezvous. For the purposes of offline evaluation, the facility to log all data distributed between game players was also provided, with NTP synchronized time stamping. Furthermore, with the evaluation performed between desktops over Ethernet, a controlled delay queue was added to the send method of the arbitrator (and the equivalent Rollback method), allowing us to emulate any end to end delay we wish between game clients.

5. EVALUATION

Evaluation of Rendezvous was undertaken over a series of latencies, each providing detailed and objective comparisons between the Rendezvous consistency mechanism and two alternative solutions: Rollback, a well-known consistency mechanism, and a control implementation with no consistency mechanism. Results were logged for a group of 4 distributed, players, with each player testing the game under the operation of each of the three consistency mechanisms at a given latency. Each game ran for approximately 5 minutes, with the results shown in Figures 4-9 reflecting typical samples from the data collected. Subsequent results reflect the entire available data set.

Latencies were chosen to reflect the target of consistency management in high latency environments, specifically latencies corresponding to 3G and GPRS latencies, as shown in Table 1. Rollback operated under a local lag of 120ms, the accepted limit of human tolerance for lag in real-time applications [1].

Table 1 Evaluation Latencies

Latency (ms)	Target
0	Baseline test
100	Common wired network delay. Just below Rollback local lag threshold.
150	Just above typical local lag threshold.
500	Typical 3G latency.
1000	Typical GPRS latency.

Additional latencies of 250ms and 750ms were also tested to provide a greater understanding of the exhibited trends in consistency at higher latency.

5.1 Consistency

In order to provide a quantified measurement of consistency in the game, a measurement of deviation between game views was necessary. In the case of Knockabout, this deviation referred to deviation between corresponding players and the ball as they moved around the pitch.

Consistency between players was measured as the deviation of the reference player's view of an object within the game with respect to the player's actual position. Ball consistency is taken as the average deviation of the ball shown by all players. In each of the following graphs, the ball is represented by the darkest line, with the 4 players shown in lighter colours.

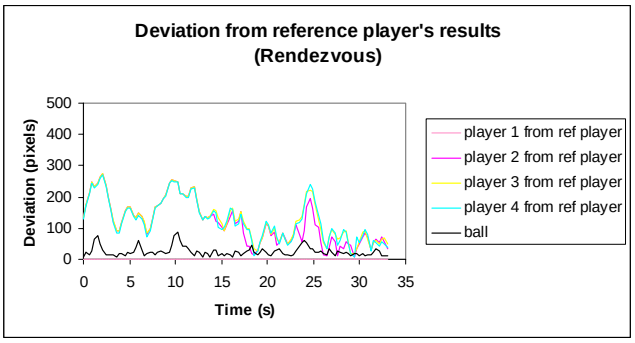

Figure 4 Consistency at 500ms in Rendezvous

Figure 4 shows the deviation between each view of a Knockabout session using Rendezvous as its consistency mechanism. Both players and the ball are illustrated as seen by one of the participating, distributed users at 500ms. Firstly, it can clearly be seen that the priority of the ball positioning over the players has been successful, with the ball shown to have a lower deviation than the players throughout the game. It can also be observed that the majority of larger deviations in player positions correlate directly with a deviation in the ball's position. This can be explained by the increased priority of the ball's position over opposing players' positions within the Rendezvous consistency mechanism. In Rendezvous, in order to maintain consistency of the ball's position, rules are put in place to ensure that should the ball deviate from its consistent target position, one of the remotely controlled players will move to intercept and return it to its intended position. As a result, any discrepancy in the position of the ball will have a

subsequent effect on the extent of one or more of the players' deviation. Finally, and critically, the close similarities between deviations of all players in Rendezvous (relative to the reference player), shown by the near precise contours of the lines representing the players in Figure 4. This interpretation, confirmed by player's experiences of the game, demonstrates the degree of semantic understanding of the game maintained through Rendezvous even at high latency.

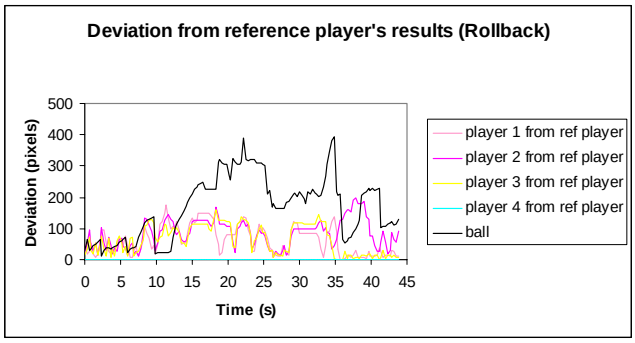

Figure 5 Consistency at 500ms in Rollback

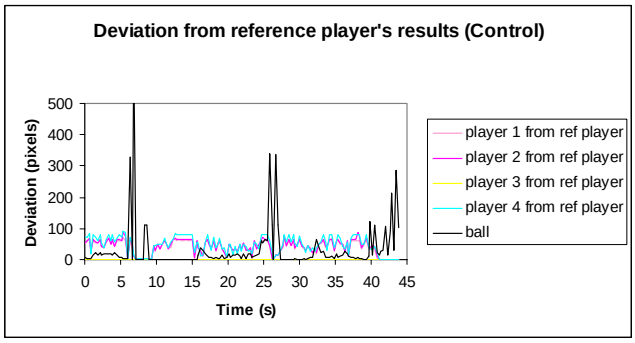

Figure 6 Consistency at 500ms in Control

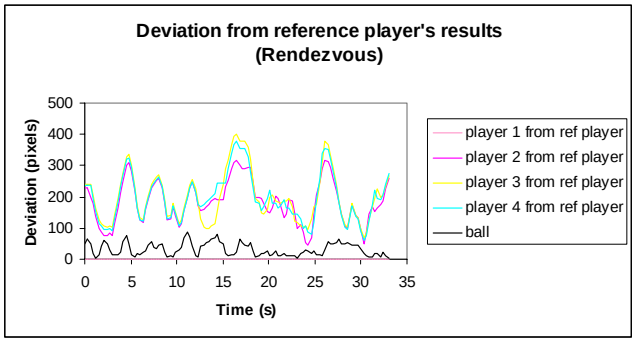

Figure 7 Consistency at low latency in Rendezvous

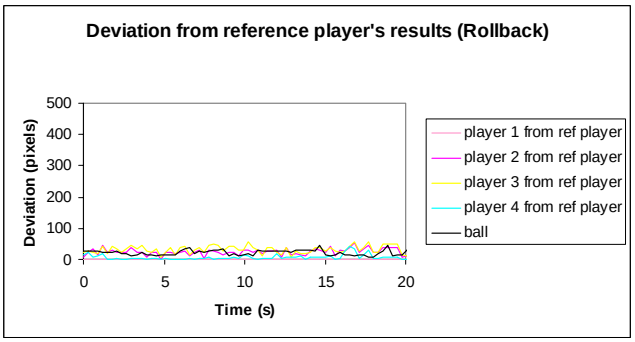

Figure 8 Consistency at low latency in Rollback

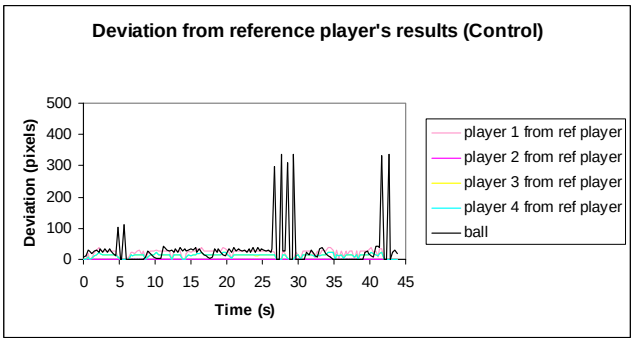

Figure 9 Consistency at low latency in Control

Taken from the games for the equivalent Rollback and control mechanisms, Figures 5 and 6 show the deviation of players and the ball at 500ms for comparison.

These three graphs (Figures 4,5,6) clearly demonstrate the difference in approach between the three mechanisms. Whereas in Rendezvous the tracking and recovery of the ball can be seen as a continuous process, Figure 6 shows the effects of objects normalising when no consistency mechanism is available, with the spikes in the graph representing large deviations in consistency followed by instant repositioning. Furthermore, although Figure 5 shows significantly less instances of objects instantly normalising, having surpassed the local lag threshold within Rollback, there are also instances where a rollback results in instantaneous repositioning of an object, in particular the ball, as demonstrated by the sharp vertical drops in the graph corresponding to rapid reductions in deviation from consistency.

Figures 5 and 6 also demonstrate that both the control and Rollback deliver increased consistency of players' positions over the position of the ball, with Rendezvous proving far more accurate with ball positioning at the cost of the players.

The difference between the three mechanisms can be further demonstrated by observing the equivalent graphs (Figures 7, 8 and 9) at low latency, enabling effective use of local lag in Rollback. Here, the changes in Rendezvous' performance (Figures 4 and 7) are minimal when compared to values at 500ms, with the frequency and scale of deviations remaining relatively constant. Consistency in the control (Figures 6 and 9) also retains the same frequency of recovery, with changes that can be attributed purely to scale as the normalising process discussed previously results in smaller but still significant sharp drops in deviation. Rollback, however, (Figures 5 and 8) demonstrates entirely changed features, with little more than deviation noise displayed.

To calculate an overall result from the initial deviations shown, each of the four players were then used as a reference player in turn, as seen in Figures 4-9, to calculate deviations of each player and the ball from every available viewpoint. These results were then taken and the mean of the deviations given calculated. The overall results given by these calculations are shown in Figure 10, below.

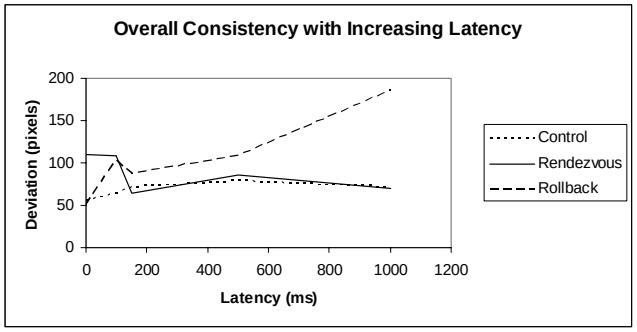

Figure 10 Overall Consistency Measurements

As can clearly be seen in Figure 10, with no delay, Rollback provides the optimal consistency, whilst at higher latencies, Rendezvous gives a far lower deviations. It is, perhaps, unexpected however that the control, with no consistency mechanism should provide the most reliable consistency, according to this analysis.

5.2 Playability

The revelation that the absence of a consistency mechanism can apparently provide better consistency than either of the opposing methods, brings into question the function of a consistency mechanism in real-time games. Naturally, a consistency mechanism is intended to provide consistency between players, but it must simultaneously enable each player to maintain a view which is consistent enough within itself to make the game continue to be playable. Results shown in Figures 6 and 9 already suggest that the instantaneous normalisation feature of the control will attribute to an increasingly unplayable game.

Subsequently, continuity within a player's own game was also calculated, taking the difference in position of each object within the game between frames (30ms) and using the probability of large, sudden movements as a measure of playability. Whilst cross-player consistency can be measured purely in terms of deviation, this measure of playability must necessarily allow for a degree of deviation between frames as the game progresses. As a result, the change in position of objects within the game, from 0 to a maximum of 1000 pixels (the size of the pitch) were calculated, and the probabilities of large changes (>150 pixels) occurring were then calculated, with Figure 11 showing the probability of large position changes for each game across each of the chosen latencies.

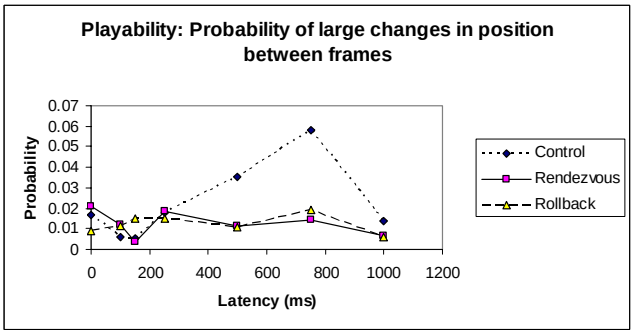

Figure 11 Playability

Here, the benefits of a working consistency mechanism become apparent. Whilst both Rendezvous and Rollback maintain playability despite increasing latencies, the control, no consistency, game rapidly becomes unplayable.

It is noticeable, however, that there is a sharp drop in the probability of large discontinuities for the control, no consistency mechanism, at 1000ms latency (Figure 11). Further investigation can reveal that at 1000ms latency the control was experiencing difficulties in maintaining consistency of players' positions, unlike at lower latency where only the ball posed any problems. This in turn gave rise to increasing, but relatively small (between 100 and 150 pixels), discontinuities in the positioning of players and as a result players found the game difficult to play. Examination of the movement of the ball throughout the game, in combination with user feedback, shows

that with rising sharp normalisation of players the game become so difficult to play that it was rare for players to succeed in kicking the ball at all, thus eliminating the far larger discontinuities seen at 500 and 750ms latency due to this mechanism's sharp normalisation of the ball.

Similarly, it should also be noted that at 1000ms Rollback had ceased rolling back to correct inconsistency between players, thus enabling each player to play independently with very little input from their opposition. This was caused by an overflow in the queue of snapshots necessary for Rollback implementation, with any potential rollback circumstances occurring proving too old to provide any data to rollback to. This discrepancy affords an interesting case for analysis, highlighting the difference between pure playability and pure consistency as here basic playability was available, but at the cost of removing any interaction from the game.

5.3 Consistency and Playability

Logically, the implication of these results is that a consistency mechanism must provide a balance of consistency and playability in order to enable an enjoyable game experience as intended. Figure 12 provides an indication of the balance provided by each of the mechanisms in question, with consistency results plotted (in terms of deviation) against playability (as measured by the probability of large discontinuities) for each of the latencies used in the experiments discussed.

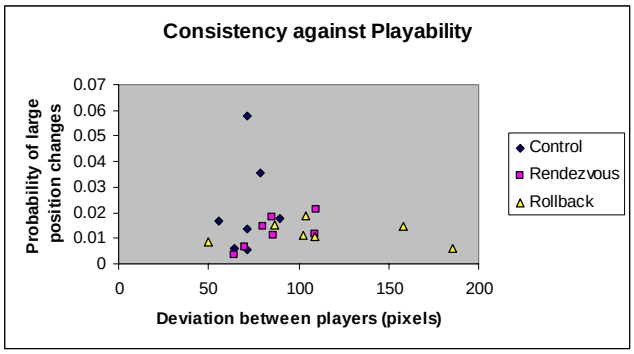

Figure 12 Consistency against Playability

As can be seen from the clustering of the points representing Rendezvous in Figure 12, increasing latency has little overall effect on either consistency or playability using this mechanism. Furthermore, the points shown on the graph would suggest a weak trend to maintain a balance between consistency and playability in Rendezvous. This can be seen more clearly in Figure 13, where the most extreme points from both the control and Rollback have been excluded.

Figures 12 and 13 also demonstrate the clear tendencies of Rollback and the control (no consistency mechanism) experiments. In each of these cases significant effects can be observed as a result of increasing latency, with Rollback maintaining playability at the cost of consistency at very high latency, and the control maintaining consistency at the cost of all playability.

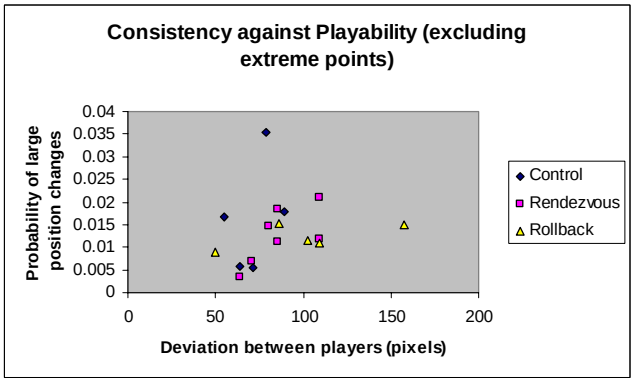

Figure 13 Consistency against Playability Central Points

To summarise, whilst at lower latencies, Rollback remains the optimal consistency mechanism, above the local lag threshold, Rendezvous' more balanced approach provides the necessary functionality without detriment to the game as a whole. As can be seen in Figures 4 and 7, the changes in Rendezvous as a result of increased latency have little effect on the consistency of the game, with playability similarly unaffected.

6. Conclusions and Future Work

Overall, we feel that Rendezvous has proved highly successful in these trials, with the detailed comparisons against Rollback confirming that the platform performs as intended and confirming suspected limitations of the Rollback mechanism. Results presented here clearly show Rendezvous to be capable of operating in high latency environments equivalent to typical GPRS latencies without detriment to its ability to maintain either playability or consistency to its targeted level.

A further comparison which is also of relevance, although beyond the scope of this paper, is the use of dead reckoning and prediction methods as a means to

combat the effects of high latency. Dead reckoning in itself does not provide a consistency mechanism, and is unable, for instance to ensure that estimates of ball position, and in particular changes in ball position, correspond to a player being in position to kick the ball. Furthermore, the method is prone to significant increases in error as latency increases [12][13], which across typical GPRS latencies may prove unusable. However, dead reckoning mechanisms have been proved to operate successfully on objects such as the players at relatively high latency (800ms) [14] and as a result are worthy of further investigation.

Perhaps the biggest question behind loose causal consistency is not its ability to maintain a degree of consistency, or, in fact, at this stage the latency at which it can perform, but rather its effect on game players. Although the playability rating goes some way to allaying fears that the game will be unplayable, qualitative user feedback is clearly an essential part of this evaluation process. Early indications, taken during these trials from the participating players, suggest that at high latency Rendezvous is indeed the most enjoyable of the mechanisms presented here, but future more detailed analysis must be completed before this can be confirmed. This work is currently underway, and is intended to not only address playability issues as applied to the soccer game discussed here, but also alternative games, including a first person shooter.

Given, at this time, feedback suggesting that the mechanism was highly successful, preliminary comments have suggested that the loosely connected causality may present problems with team play within the game. To some extent this will be inevitable at high latency, however, in the subsequent phase of the Rendezvous project work will be carried out towards reducing the effects of the mechanism on team game play, and equivalent alternative applications. This work will be focused on the development of prediction and profiling aspects of the platform, and it is expected will significantly improve the ability of players to interact.

Rendezvous will also further need to prove itself under increasing numbers of players. Results presented here discuss a four player game, but preliminary trials with eight players have already been carried out. These trials suggest that with increasing numbers of players, the peer to peer nature of the game will place far too much strain on the system bandwidth, particularly if operating over 3G or GPRS. To this end, the incorporation of a relevance filtering mechanism, LUCID [8], as a means of bounding the utilised bandwidth is already underway. The LUCID filter uses spatial partitioning [10][11] and area of interest filtering to ensure that game players only exchange information with those to whom it is relevant (in this case the closest players), and it is anticipated will enable significant saving in overall bandwidth. Further work is planned to explore any potential interactions between the Rendezvous mechanism and the effects of area of interest filtering.

Finally, in deference to Rendezvous' roots in control theory, we anticipate longer term future work focusing on the use of more complex control theory notions beyond the relatively simple proportional control put into use here, thus improving the overall response.

References

[1] Pantel L, Wolf LC, On the impact of delay on real-time multiplayer games. *Proceedings of the 12th International Conference on Network and Operating Systems Support for Digital Audio and Video, Miami, Florida 2002,* pp 23-29.

[2] Chandler A, Finney J, Rendezvous: Supporting Real-time Collaborative Gaming in High Latency Environments. *Proceedings of the Second ACM International Conference on Advances in Computer Entertainment Technology (ACE 2005) in co-operation with SIGCHI,15-17 June 2005, Valencia, Spain.*

[3] Prakash A, Knister MJ, A framework for undoing actions in collaborative systems. *ACM Transactions on Computer-Human Interaction (TOCHI) Vol 1, Issue 4, Dec 1994,* pp 295-330.

[4] Mauve M, How to keep a dead man from shooting. *Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS), 2000*, pp 199-204, Enschede, Netherlands, October 2000.

[5] Jefferson D, Virtual Time. *ACM Transactions on Programming Languages and Systems, July 1985*

[6] Jefferson D, Virtual Time II: Storage Management in Conservative and Optimistic Systems. *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing, Quebec 1990,* pp 75-89.

[7] Armitage G, Sensitivity of Quake3 players to network latency. *Proceedings of Internet Measurement Workshop Poster Session (IMW), 2001*.

[8] McCaffery D, Finney J, The need for real-time consistency management in P2P mobile gaming environments. *Proceedings of the First ACM International Conference on Advances in Computer Entertainment Technology (ACE 2004) in co-operation with SIGCHI, 3-5 June 2004*, Singapore.

[9] Frecon E, Stenlus M, DIVE: A Scalable Network Architecture Environment for Distributed Virtual Environments. *Distributed Systems Engineering Journal (Special Issue on Distributed Virtual Environments), September 1998, Vol 5, No 3*, pp 91-100.

[10] Macedonia MR et al, NPSNET: a multiplayer 3D virtual environment over the internet.. *Proc of the 1995 symposium on Interactive 3D graphics, Monterey, California, USA, 1995*.

[11] Macedonia MR et al, Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. *Proc of the virtual reality annual symposium (VRAIS 95), IEEE Computer Society, 1995*, pp 2-10

[12] Pantel K, Wolf LC, On the Suitability of Dead Reckoning Schemes for Games. *Proceedings of 1st ACM SIGCOMM Workshop on Network and System Support for Games, (Net Games) April 2002*. pp 79-84.

[13] Borenstein J, Internal Correction of Dead Reckoning Errors with a Dual-Drive Compliant Linkage Mobile Robot. *Journal of Robotic Systems, 1995, Vol 12, No 4*, pp257.

[14] Aggarwal S, Banavar H, Khandelwal A, Mukherjee S, Rangarajan S, User Experience: Accuracy in Dead-Reckoning Based Distributed Multi-Player Games. *Proceedings of 3rd ACM SIGNCOMM Workshop on Network and System Support for Games (Net Games) 2004*. pp 161-165.