Rendezvous: The Case for a Highly Optimistic Real-Time Consistency Mechanism

Angie Chandler, Joe Finney, Computing Department, Lancaster University, UK
email: {angie, joe} @comp.lancs.ac.uk

Abstract- **The use of distributed real-time applications in mobile environments is growing. Mobile distributed computer games, distributed virtual environments and even collaborating mobile robots all require consistent shared state between real-time collaborating components, despite the harsh network conditions they are expected to face. Up until now, support for real-time collaboration over unreliable networks with high latency and frequent disconnection has been relatively scarce.**

This paper introduces Rendezvous, a real-time, peer to peer consistency mechanism intended for situations where network latencies are high and disconnections are frequent - conditions that render traditional consistency mechanisms unsuitable. Rendezvous aims to enable the nodes of a collaborating system to accept occasional temporary inconsistency in return for improved performance, and to facilitate the gradual convergence of conflicting collaborators back to a consistent state. This paper also discusses the requirements essential to the design and implementation of such a consistency mechanism, the reasoning behind these requirements, the challenges that must be faced, and their application to specific examples of high latency, frequently disconnected devices.

***Index Terms*— consistency, real-time, latency, disconnection, collaboration, mobile**

1. Existing Support for Consistency in Real-Time Collaborative Systems

Current mobile systems do not adequately support the needs of real-time collaborative applications. Moreover, systems that specifically try to support real-time collaborative applications do not perform well in mobile environments where network outages are commonplace and network latencies are extremely high. The Rendezvous project aims to resolve these issues by developing a software platform that will expose a novel programming paradigm to these applications, allowing them to operate despite the harsh network conditions found in mobile environments. Execution over unpredictable networks proves devastating to all but the most resilient of real-time applications; real-time applications are inevitably forced to continue with their execution whilst still waiting for potentially essential data from one or more of its collaborators, leading to loss of consistency throughout the system.

By far the most commonly used method for conflict resolution in these situations is based on the concept of rollback [1][2] – the original example of which being TimeWarp [3][4], an optimistic rollback mechanism that upon detection of a conflict will take the application back in 'virtual time' until the conflict no longer exists, subsequently rebuilding an acceptable sequence of events to replace it. TimeWarp records all events received but behaves optimistically – assuming that events will arrive in the correct order until proven otherwise and then rolling back to resolve the conflict, thus affecting delays as little as possible.

In order to implement a rollback mechanism in real-time systems, a degree of delay must be artificially inserted as a buffer zone, during which events are essentially unconfirmed. Taking into account that in situations where there is a degree of user interaction with the distributed application, such as distributed multiplayer games or distributed virtual environments, this delay can cause serious problems. Even across a reliable network environment the delay between nodes on the internet is expected to be of the order of 200ms, already close to the well documented maximum tolerable delay for the user of 250ms [5][6].

Once network traffic is moved into mobile environments, however, latencies can increase dramatically. Whilst high performance local area networks such as IEEE 802.11 are theoretically almost as fast as wired networks, with delays of the order of a few milliseconds, in ad hoc situations performance can be severely affected by network topology, and they remain prone to interference, contention and packet loss; the latter often introducing further delays due to the retransmission requirements of TCP or other such reliable transmission protocols. Furthermore, mobile devices must often continue to rely on wide area networks, such as GPRS or its 3G replacements, i.e. wide band CDMA and EDGE. These networks exhibit significantly higher latencies for much of their transmission. GPRS, which is currently in use, has expected latencies in the range of 1000-2700ms (round trip time) [7] with the agreed specification setting the latency at 800ms and above with a maximum of 10 seconds delay [8], and whilst the 3G systems improve on this, it is clear that these wireless systems will remain prone to delays for the foreseeable future.

Of course, the difficulties inherent in mobile distributed systems aren't limited to network latency, it is well understood that all mobile networks, from ad hoc networks to the most reliable GSM infrastructures, are prone to disconnection. These disconnections can be caused simply by moving out of range of the wireless infrastructure, or by partitioning in ad hoc situations. Disconnection is further complicated in environments where power conservation is of importance, such as sensor networks and mobile robotics [9]. Here, disconnection can be caused in an ad hoc network simply because a bridging member of the network needed to conserve energy. Moreover, it is also this type of application, in particular robotics, where the rollback mechanism becomes increasingly problematic. Were rollback to be applied to a team of collaborating robots it may become necessary for a robot to literally, rather than virtually, retrace its steps, an action which may not always be possible for applications which operate in the physical world.

In view of the necessary limitations placed on the acceptable delay between users of collaborative distributed applications, and further difficulties presented in the form of frequent disconnection in mobile distributed systems, it is readily apparent that rollback mechanisms as they stand will prove inadequate in high latency, frequent disconnection situations. The following section and remainder of this paper propose an alternative solution and the challenges this alternative must face.

2. Rendezvous

Given the likelihood that mobile and wireless networks will continue to be high latency and error prone, the need to operate real-time applications in this environment, and that existing rollback mechanisms would be inappropriate as a means of maintaining shared state consistency, it is clear that an alternative solution has become necessary. The approach proposed in this paper, Rendezvous, suggests allowing the controlled sacrifice of a degree of accuracy or consistency in favour of performance. Certain parallels with this approach can be drawn from fault tolerant applications, where a subset of highly fault tolerant applications can use rollback, whereas others require redundancy. For example, both electronic financial transactions, as performed by a bank, and the control systems behind an aeroplane can be regarded as highly fault tolerant, but given the differing nature of the operations performed only the financial transactions can make use of rollback when an error occurs. As an alternative, the aeroplane instead has to incorporate redundant systems, providing a fall back mechanism when one fails [10]. In Rendezvous too a level of redundancy is introduced by the use of independently functioning collaborating nodes, each of which optimistically maintains their own, marginally inconsistent, world view. This parallel is no coincidence; rather it reflects the similarity of the problems faced by these two independent areas, with the two extremes of perfect consistency and perfect performance forming a scale upon which applications must choose the appropriate balance.

Rendezvous is a consistency mechanism which we like to term *temporally monotonic*, intended for situations where rolling back virtual time would be inappropriate but where intermittent inaccuracies in consistency are tolerable. The mechanism is focussed on the gradual recovery from errors in consistency rather than their absolute prevention. Here, when an inconsistency occurs, the Rendezvous mechanism will calculate a future point in time and space where the two inconsistent elements of the application's shared state can once again become consistent by allowing the collaborating entities to gradually converge. This convergence can be achieved by subtle bending of the rules of the environment, ensuring that seamless recovery from the inconsistency is possible in the majority of cases.

To ensure that the general concept was sound, a few initial experiments have already been carried out on a distributed form of the classic computer game Pong [11]. These experiments showed a simple version of the Rendezvous mechanism enabling two players to continue to fairly and effectively play the game Pong even with a delay of 2800ms between them. These initial results, when compared to the maximum tolerable delay under traditional mechanisms of approximately 250ms and the anticipated delays over GPRS of the order of 2700ms, clearly showed this approach to be worthy of further investigation, although further details of this experiment are beyond the scope of this paper.

3. Challenges

Moving beyond the simplistic Rendezvous mechanism applied to Pong, there are a number of challenges to be met in order to successfully achieve full, generic implementation capable of supporting complex real-time applications. The Rendezvous mechanism is composed of a series of fundamental stages, each related to the detection of and recovery from an inconsistent shared state. These stages, from detection to reconciliation, are supported by the underlying structure of the system. Specifically, each collaborator within the group must maintain a *profile* of every member of the group, including themselves. These profiles maintain the estimated state of collaborators based on a limited range of essential data, such as position information on a mobile robot. Profiles form the underlying support for the detection of inconsistency in the state. This allows any probable inconsistency between expected (profile) and genuine action to be detected by the member undertaking the action as quickly as possible, even in an unreliable network environment. Any node detecting a divergence of state then becomes responsible for calculation of a target state, based on its knowledge of those it is in collaboration with.

However, before the challenges brought about by the proposed implementation of Rendezvous can be considered, certain assumptions must also be taken into account. Firstly, Rendezvous is intended to be implemented on a fully distributed, non-centralised system. Secondly, communications are assumed to be reliable, based on use of a reliable transport protocol such as TCP, with delays for any retransmissions taken into account as part of the consistency mechanism. It is also assumed to have a means of clock synchronisation in place, although this could in itself become part of the Rendezvous mechanism at a later stage.

3.1. Profiling

In order for Rendezvous to be optimistic, it is essential for there to be a form of prediction to enable disconnected or delayed collaborators to approximate the actions of the other members of their group and maintain their virtual state. As the profiling of the delayed/disconnected collaborators improves, the chances of an inconsistency between the disconnected collaborators will decrease, thus decreasing the frequency at which reconciliation will need to be performed and improving the functionality of the system as a whole.

Accurate profiling is difficult to achieve, and in the case where collaborators are disconnected for any length of time, the profiling will inevitably become increasingly erroneous. For the purposes of Rendezvous, which must also limit communications even during adequate network connection in deference to the power needs of the majority of mobile systems, a select few variables will be chosen as *profile variables* to be shared between all interested collaborators whenever possible. These variables will allow them to maintain expected profiles of every other member's state.

As can be seen in Figure 1, each collaborating node maintains the profile, and related state, of each of its collaborators and also of itself, with the application state completely separated from any processing. The Rendezvous mechanism, transparently applied between the state and application, will detect changes in profile variables within the real state of the collaborator and notify other collaborators accordingly.

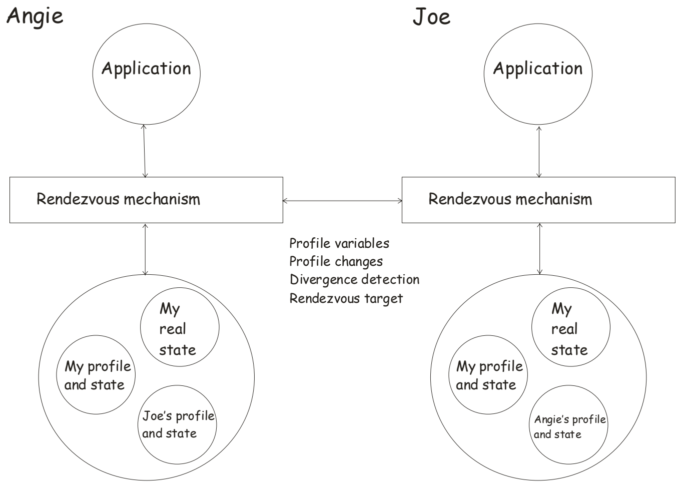

Figure 1 Two Collaborating Nodes

3.2. Detecting divergence

Naturally, for the Rendezvous mechanism to be triggered there must first be a means of detecting inconsistency, or *divergence* of the shared state. This process is made doubly difficult by the inevitable network problems associated with this event, as divergence is most likely to occur when there is already a current serious network latency or disconnection.

However, with the use of profiling on each of the separate collaborators, and specifically the profile each collaborator will maintain of itself (see Figure 1), this process will be significantly accelerated as each collaborator is made responsible for detecting its own likely divergences. These divergences will be detected by checking for differences in a few *critical variables* between a collaborator's real situation and the current predicted state from the collaborator's profile of itself, thus enabling it to notify all concerned parties as soon as a potential divergence is detected.

To clarify, the exact nature of a critical variable may ultimately operate on a sliding scale, dependent on the requirements of the application and its environment. However, in its simplest form, as for example in the game Pong, it may be taken to represent the status of the ball, specifically whether or not the ball is currently in play. Should a collaborator detect a conflict in this variable, ball status, between its two models of itself – one real and one profile – it would then notify others of a divergence.

3.3. Choosing and Reaching a Rendezvous target

Once a divergence of shared state has been detected, the collaborators must then decide on a convergence point which is acceptable to all parties in order to begin the convergence process. This calculation of a convergence target or *Rendezvous target* requires a degree of understanding of the underlying application so that it can be assured that all conflicting entities are able to reach it. Further to this, it is possible that some or all of the collaborators will already be in the process of converging to a previous target state based on a previous divergence, so the new Rendezvous target must take this into account.

Following on from the detection of the initial divergence, from an efficiency perspective it is clear that the creator of the divergence should also be the one to calculate the convergence point, sending the proposed solution to its collaborators along with the specification of the original mistake. This approach will also ensure that only one convergence target is suggested, avoiding the need for an extended period of communication or assumptions about the synchronisation of each profile in each collaborator. Should any of the informed collaborators disagree on the Rendezvous target this will in essence signify a second parallel divergence and inaccuracy in the original collaborator's model, leading to a second calculation of an improved Rendezvous target and a second notification of the group. In the meantime, all other collaborators will have assumed that the first Rendezvous target was accurate and optimistically begun convergence on that assumption.

The Rendezvous target itself is chosen through the application of *rules* beyond those within the application itself to enable the manipulation of the application state outside its normal functionality or environment. For instance, in the gaming world, this bending of rules might include artificial encouragement of a player to head in a certain direction, or simulation of an element of the environment even beyond its normal tolerances. These rules are applied in order to enable a smooth convergence of conflicting states, but currently stand as the largest single challenge within the proposed Rendezvous mechanism, requiring a high degree of meta-understanding of the application.

3.4. Challenge Summary

As work progresses on the implementation of the challenges discussed in this paper, it is anticipated that further challenges, such as a need for fairness in competitive collaborative situations, will continue to present themselves. However, despite the challenging nature of some areas of this work, we strongly believe there is currently sufficient need for the work and solutions to the problems to be faced will be found through continued investigation.

4. Ongoing work

As with any work of this nature, it is essential that a variety of alternative applications be put into use in order to facilitate a more complete and unbiased understanding of the associated problems. Following on from our initial experiments using a distributed version of the game Pong, we are currently undertaking the implementation of a small team of collaborative robots. Initial experiments with these robots may only feature a single robot as it interacts with the environment, providing a base line experiment from which further details on the precise requirements of critical and profile variables can be ascertained, and furthermore the degree to which Rendezvous rules can be allowed to bend the rules by which a robot will normally abide.

As indicated, our current thinking suggests that the best way to progress with a Rendezvous implementation requires concentration on the profiling and maintenance of state aspect of the mechanism, prior to the investigation of Rendezvous reconciliation rules, primarily due to the inter-dependencies of these two complementary challenges. Firstly, to maintain the collaborators profiles each collaborating node will be required to distribute any changes in its profile variables to other members of the group. To enable Rendezvous to operate transparently to the application, the Rendezvous will monitor for changes in profile variables in the relevant state and distribute the information.

Secondly, the changes observed must also be used to maintain profile accuracy, so we propose that each time a profile variable is updated by the Rendezvous mechanism a *profile rule,* in essence a rule which defines how profile variables change over time, will be added or updated to reflect the nature of the change. Later, when a divergence is detected that implicates an error in these rules they may be forcibly updated, with the creator of the divergence using detailed knowledge of its real situation to calculate an improved profile and share that information.

5. REFERENCES

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

[2] How to Keep a Dead Man from Shooting, Martin Mauve, Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS), 2000, pp199-204, Enschede, Netherlands, October 2000.

[3] Virtual Time, David Jefferson, ACM Trans on Prog. Lang. and Sys., July 1985.

[4] Virtual Time II: Storage Management in Conservative and Optimistic Systems, David Jefferson, Proceedings of the ninth annual ACM Symposium on Principles of Distributed Computing, Quebec 1990, pp75-89

[5] On the Impact of Delay on Real-Time Multiplayer Games, Lothar Pantel, Lars C. Wolf, Proc of the 12th International on Network and operating Systems support for digital audio and video, Miami, Florida, 2002, pp 23-29

[6] Sensitivity of Quake3 Players to Network Latency, G. Armitage, Proc of IMW 2001 Internet Measurement Workshop Poster Session, 2001

[7] Cambridge Open Mobile, Ian Pratt, 2001, http://www.acu.rl.ac.uk/msn2001/talks/Ian_Pratt.pdf

[8] Flarion's Tokyo Wireless Adventure, Guy Kewney, The Register, 2004, http://www.theregister.co.uk/2004/05/07/flarion_tokyo_adventure/

[9] Data Gathering Algorithms in Sensor Networks Using Energy Metric, S. Lindsay, C. Raghavendra, K.M. Sivalingam, IEEE Transactions on Parallel and Distributed Systems, Vol 13, No 9, September 2002, pp 924-934

[10] A Design Approach for Ultrareliable Real-Time Systems, J.H.Lala, R.E.Harper and L.S.Alger, Readings in Real-Time Systems, eds Yann Hang Lee and C.M.Krishna, IEEE Computer Society Press, 1992, pp 3-30

[11] The Story of Pong: Ralph H. Baer, Inventor of the video game, David Winter, 2004, http://www.pong-story.com