Managing latency and fairness in networked games

J. Brun
University of Wollongong

Farzad Safaei
University of Wollongong, farzad@uow.edu.au

P. Boustead
University of Wollongong, boustead@uow.edu.au

Managing latency and fairness in networked games

Abstract

Networked games can be seen as forerunners of all kinds of participatory entertainment applications delivered through the Internet. Physically dispersed players are immersed in a common virtual environment where they interact in real time. When a user performs an action, other users must be made aware of that action. Otherwise, there is a discrepancy in the perceptions of participants about the overall state of the virtual world. This discrepancy could lead to undesirable and sometimes paradoxical outcomes. In particular, first-person shooter, and to a lesser extent role-playing games impose stringent constraints on responsiveness and consistency.

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Brun, J, Safaei, F & Boustead, P, Managing latency and fairness in networked games, Communications of the ACM (CACM), 2006, 49(11), 46-51. Copyright 2006 Association for Computing Machinery (ACM). The journal webpage is available here.

Managing Latency and Fairness in Networked Games

Jeremy Brun, Farzad Safaei and Paul Boustead
Smart Internet Technology Cooperative Research Center,
University of Wollongong, Australia
Email: farzad@uow.edu.au, {jeremy,paul}@titr.uow.edu.au

Introduction

Networked games can be viewed as forerunners of participatory entertainment applications delivered over the Internet. In these games, physically dispersed players are immersed in a common virtual environment and interact in real time. When one user performs an action, other users should become aware of this. Otherwise, there will be a discrepancy between the perceptions of participants about the state of the virtual world. This may lead to undesirable and sometimes paradoxical outcomes. In particular, the First Person Shooter (FPS) category, and to a lesser extent Role Playing Games (RPG), have stringent constraints on responsiveness and consistency.

Consequently, deploying these applications over a large-scale infrastructure presents a significant challenge. As the geographical distances between the participants increase, the unavoidable propagation delays between the participants may render the game unresponsive and sluggish even when there is abundant processing and network resources available. Furthermore, the differences in game responsiveness to user inputs can provide some players with unfair advantages.

To limit the impact of these problems, most games are currently deployed as independent virtual worlds for localized areas and served by machines dimensioned for peak hour demand. However, the true power of these applications is to enable people to work and play together irrespective of physical separation. Confining games to small localities is analogous to having a telephone network that can only handle local calls. Nevertheless, geographical scaling of network games is non-trivial and involves much more than providing network connectivity and bandwidth.

This article presents an overview of various factors that can affect the quality of the game experience in terms of playability and fairness. In software, different methods for synchronization and lag compensation can reduce the perceptual impact of latency. Furthermore, it is demonstrated that careful selection and organization of game servers can be of significant value in improving playability and fairness.

Anatomy of network games

At any time, the virtual world of the game is fully described by a set of parameters called the *game state*. Such parameters include, but are not limited to, the position and states of avatars and other in-game objects. Players perceive and react to the game through their *terminal*, a networked computer or game console which renders the virtual environment based on game state updates. Authoritative modifications of the shared game state are done by *decision points* located either on dedicated servers or on (a subset of) players' machines.

There are two main architectures for making decisions about the game state: *central* or *distributed* decisions points. Most current networked games use a *central server* architecture, where a single machine is the unique decision point. In *distributed* architectures, two or more decision points coexist and must synchronize their game state with each other.

The need for synchronization in the distributed architecture introduces additional complexity and if extra dedicated hardware is required, it adds to the running cost for the game provider. On the other hand, distributed architectures, such as mirrored servers and peer to peer games, provide more flexibility for load balancing and may improve player's experience if the decision points are located close to participants.

Playability and fairness

Geographical distance between decision points and/or terminals may cause their respective states to become somewhat inconsistent with each other because of the latency involved in the transmission of information. Two classes of inconsistencies can be identified: inconsistencies between a terminal and its relevant decision point(s) and inconsistencies between different decision points. The latter only applies to the distributed architecture where there is more than one decision point.

One example of inconsistency caused by the propagation delay between a terminal and its decision point is the *response time*. The response time represents the delay between the time of the issuance of an action order by a player and the rendering of the action results on the player's terminal. A perceptible response time is frustrating for users and beyond a certain threshold the game may not be playable. A second example is the *presentation inconsistency* [10] which is due to the fact that the game state update reaching a terminal is already outdated to some degree because the real game state may have varied while the update packet was on its way. Hence, what the player perceives is slightly inconsistent with the real game state at the decision point. This may cause a player to see other avatars at incorrect locations, for example.

The distributed architecture may help reducing response time by bringing decision points closer to players. However, it introduces inter-decision point inconsistency, that is, discrepancies *between* different decision points with respect to the 'current' game state. This can cause some decision points to evaluate actions out of order, possibly violating causality and taking incompatible decisions. A *paradox* is a decision made by an inconsistent server which is incompatible with the decision it would have made if it was consistent. Paradoxes arise only due to the discrepancy between decision points and cannot happen in the central server model. If causality is to be maintained, paradoxical game states have to be healed by rolling back to an earlier time point. This is called a *roll back* in time, also referred to as a *Timewarp* [5].

A game can be considered to be *playable*, provided the users find the performance of the game acceptable in terms of the perceptual impact of different inconsistencies. Whereas playability is a game attribute for each individual player and varies depending on the inconsistencies that the player experiences from his/her terminal or decision point, *fairness* is a game wide property concerned with *relative* playability *amongst* the participants [3]. In other words, variations in playability between players may give unfair advantages to some over others. If these variations are significant, the game may be considered to be unfair. A fair game, on the other hand, would provide all users with the same level of handicap.

Aside from artistic design and originality, the quality of online game experience critically depends on the network aspect of playability and fairness. This is why management of network related inconsistencies in the supporting game infrastructure becomes crucial when scaling to wide geographical areas. Different techniques to achieve this outcome are presented next.

Tuning the game infrastructure

The infrastructure supporting a networked game can be divided into a software component including the synchronization scheme and lag compensation techniques, and the hardware infrastructure which is the topology of the game decision points over the underlying network platform. The other parameters that also influence players' experience, namely the underlying network topology itself and the location of players, are not controllable. Therefore, a game provider can manage playability and fairness at two different levels:

- Trading inconsistencies within the software component. For a given topology of decision points, the network delay between entities is bounded. However, it may be possible to *trade* one type of inconsistency with high perceptual impact for another that has a lower impact. This may result in an overall improvement of game's quality from the users' viewpoint.
- Selecting the decision point topology. This way, the latency constraints which depend on the location of the decision point can be altered to influence participants' playability and fairness.

Trading Inconsistencies

The artifacts of inconsistencies, such as long response time or a large number of rollbacks, have different perceptual impacts on users. For example in the Unreal Tournament first person shooter game, Quax et al. [8] conclude that a round trip delay (response time) above $60ms$ seriously disturbs players. Likewise, it is reasonable to assume that roll backs also degrade playability.

While the latency between terminals and decision points are bounded by the propagation delay constraints of the given topology, it may however be possible to *trade* one type of inconsistency for another. For example, to reduce the response time, terminals can use *co-simulation* to anticipate the decisions made by the decision points. For actions originated by the player, this is referred to as *client-side prediction* [1] and reduces the perceived response time. The state of *other* avatars may be anticipated by *dead reckoning* [7] using knowledge about the previous values of a given parameter, such as location and direction of movement of other avatars, and physics of the virtual universe.

In both cases above, if the predicted parameters are the same as the authoritative updates received from the decision point, then the perceived response time or presentation consistency are significantly improved. There is a probability, however, that the authoritative decisions would have to *revoke* the local predictions if they were incorrect. The revocation may be perceived by the player as a local roll back. In essence, these techniques trade improved response time and presentation consistency for an increased probability of revocation. This trade-off may or may not be appropriate depending on the context of the game and the perceptual impact of each inconsistency type. The above methods, referred to as *lag compensation techniques*, are concerned with hiding the impact of inconsistency between terminals and decision points and apply to both central and distributed topology.

The distributed architecture introduces inter-decision point inconsistency and the possibility of paradoxes. One may adopt a *conservative* synchronization scheme between the decision points which eliminates the probability of paradoxes altogether. Examples include conservative *local lag* [5] or lock-step synchronization [4]. Such schemes, however, would affect the game's responsiveness, negating some of the benefits of using distributed architecture. On the other hand, a more *optimistic* synchronization scheme may be used. In this approach some level of inconsistency between the decision points is allowed and it may be essential to heal a paradoxical game state using roll backs. Once again this approach trades one type of inconsistency for another. For example, Figures 1 illustrates how inter-decision point inconsistency can be traded for an increase in response time by adding local lag, assuming no packet loss or jitter. Partial local lag can also be used to reduce, without fully eliminating, the duration of the inter-decision point inconsistency. The longer this duration, the higher the probability of paradox. Hence, it might be worthwhile to set the local lag such that the optimal balance between the perceptual impact of response time and roll backs is attained.

Different parameters of the virtual world may represent totally different in-game concepts which may have different consistencies and synchronization requirements [2] [9]. As an example, in most online RPG, an error in the avatar's position would not typically affect the actions of other participants due to limited acceleration and speed. Yet, players would want to see their avatars reacting quickly once they have decided to move. On the other hand, a paradox on an avatar's life state -dead or alive- may have significant negative impact on the game playability. Therefore, actions affecting avatar's positions could use less local lag than actions affecting avatar's life state. In general, it could be more effective to tailor the synchronization parameters for each action type rather than binding the whole game state to the same synchronization fate.

It is always possible to *increase* the level of inconsistencies in a game by artificially delaying information. This technique enables the equalization of inconsistencies amongst players effectively improving the game fairness at the cost of overall playability.

Selecting the decision point topology

The physical topology of telecommunication networks is generally static. However, if we assume having access to a network of processing locations, this would provide a pool of possible decision points to choose from [6]. Under such circumstances, the position of the unique decision point for a central server model could be selected to best suit the current connected players. This selection could be based on different objectives, such as optimizing the average playability, the global fairness or a trade off between the two.

Alternatively, a distributed decision point architecture composed of a subset of carefully selected processing locations tuned with suitable synchronization parameters could provide even a better trade off over a central server solution.

In any game, the worse off player in terms of playability could be a major contributor to the game's average response time and unfairness. The response time of this player is denoted as the *critical response time*. We have developed an iterative heuristic which converges towards a set of servers with close to optimal playability for the worse off player in the game, therefore providing a balanced solution between overall playability and fairness. The synchronization scheme considered by this heuristic is a fully conservative local lag, assuming no jitter or packet loss, which implies that the response time is a good indicator of playability. An absolute lower bound for the critical response time can always be calculated, giving a measure of the quality of the obtained solution. After running this heuristic for one hundred times over a simulated Internet-like network topology composed of 600 nodes with 48 randomly position players, the average gap between the final solution and the lower bound is found to be around 5% and the number of decision points in the solution required for the final solution is about 7.5 on average.

Figure 2 shows one representative instance of the iterative evolution of the heuristic solution in terms of critical response time compared to the lower bound and the critical response time of two other selection strategies:

- Best central server in terms of average response time which purely optimizes the overall game playability.
- Best central server in term of critical response time which finds a balance between playability and fairness.

As described in Table 1, at each step the heuristic finds the worse off player and searches for the best new decision point to be added to the current server list that would reduce his/her response time. When further improvement of this critical response time is no longer possible, the heuristic ends. As can be seen, the critical response time of the heuristic, even for only 6 distributed decision points is very close to the lower bound and is significantly better than even the best central server solution. In other words, a properly designed distributed architecture is likely to outperform the current central server models under a wide range of conditions.

Figure 3 presents the performances of the three decision point topology selection strategies in the same simulated network. The horizontal and vertical axes represent the level of playability and fairness respectively; the closer to the origin the better. Each of the 100 simulations of three selection strategies generated solutions that are represented as a single point on the Figure. The combination of these points creates distinct clouds. The fourth cloud, labeled Average Central Server, represents the expected playability and fairness of a randomly chosen central server for comparison.

The central server solution chosen for optimal playability provides consistently low response time with a high level of unfairness. The outcome of the balanced central server solution is more variable: sometimes close to best playability and other times with inferior response time but an improved fairness. The distributed solution from the heuristic is consistently better than the other two strategies in terms of fairness at the cost of a slight increase in response time compared to the optimal playability server. All these selection strategies offer a considerable improvement over the expected playability and fairness of a randomly chosen central server.

Conclusion

Trading inconsistencies and tuning the decision point topology are the two available strategies to manage playability and fairness in online games. Ideally, all these techniques should be implemented in real-time to constantly adapt to the dynamics of the game and players' connections.

However, current software and hardware platforms provide little support for cost effective deployment of these capabilities on large scale. This is the reason why most games currently use a fixed central server approach in combination with some form of latency compensation, leaving room for improvement in the future.

References

[1] W. Bernier. Latency compensating methods in client/server in-game protocol design and optimization. In *Game Developer Conference (GDC)*, http://www.gdconf.com/archives/2001/bernier.doc, Mar 2001.

[2] J. Brun, F. Safaei, and P. Boustead. Distributing network games servers for improved geographical scalability. *Telecommunication Journal of Australia (TJA)*, 55(2):23–32, Autumn 2005.

[3] J. Brun, F. Safaei, and P. Boustead. Fairness and playability in online multiplayer games. In *IEEE workshop on Networking Issues in Multimedia Entertainment (NIME) at the Consumer Communications and Networking Conference (CCNC)*, Jan 2006.

[4] B. D. Chen and M. Maheswaran. A fair synchronization protocol with cheat proofing for decentralized online multiplayer games. In *IEEE symposium on Network Computing and Applications (NCA)*, pages 372–375, 2004.

[5] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and timewarp: Providing consistency for replicated continuous applications. *IEEE Transactions on Multimedia*, 6:47–57, Feb 2004.

[6] T. V. Nguyen, F. Safaei, P. Boustead, and C. T. Chou. Provisioning overlay distribution networks. *Elsevier Computer Networks*, 49:103–118, 2005.

[7] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In *ACM workshop on Network and system support for games (NetGames)*, pages 79–84, apr 2002.

[8] P. Quax, P. Monsieurs, L. Wim, D. De Vleeschauwer, and N. Degrande. User experience: Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. In *ACM workshop on Network and system support for Games (NetGames)*, Aug 2004.

[9] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun, and M. Dowlatshahi. Latency-driven distribution: infrastructure needs of participatory entertainment applications. *IEEE Communications Magazine*, 43(5):106–112, May 2005.

[10] I. Vaghi, C. Greenhalgh, and S. Benford. Coping with inconsistency due to network delays in collaborative virtual environments. In *ACM symposium on Virtual Reality Software and Technology (VRST)*, pages 42–49, Dec 1999.

NOTES FOR THE EDITOR(S)

Description of the heuristic (to be put in a table if necessary)

Caption: “The heuristic starts from an initial two-server solution optimized for the two most distant players. This solution is then expanded at each iteration to minimize the response time of the worse off player by adding a new server. The process ends when no more improvement is possible.”

- Initialisation:
  - Create a *restricted players list* composed of the two most distant players in the network.
  - Initialize the initial server list solution with the two servers providing the best critical response time to the game *restricted* to these two players alone.
- Step 1:
  - Test the current server list solution with all players
  - If no player outside the restricted player list becomes critical: the heuristic ends
  - Add the new critical player to the restricted player list.
- Step 2:
  - Look for the server which minimizes the response time of the new critical player when added to the current server list.
  - If the addition of a server no longer improves the critical response time: the heuristic ends.
  - Add this new server to the server list
  - re-iterate to Step 1

Captions for the figures:

Figure: brun.figure1.eps

Caption: “Examples of synchronization in distributed server architecture. In optimistic synchronization, the sync message from server 2 can create a paradox on server 1 if it conflicts with Player A’s action. The local lag compensates the inter-decision point inconsistencies assuming there is no jitter or packet loss.”

Figure: brun.figure2.eps

Caption: “Iterative evolution of a typical heuristic convergence. The critical response time is improved each time a server is added to the game. The final solution outperforms any central server approach and ends up close to the calculable lower bound.”

Figure: brun.figure3.eps

Caption: “Playability and Fairness of different decision point selection strategies. The set of distributed servers chosen by the heuristic outperforms the best central server approaches in terms of fairness at a marginal cost in average response time.”

Biographies:

Jeremy Brun (jeremy@titr.uow.edu.au) is a Ph.D candidate in the Centre for Emerging Networks and Applications at the University of Wollongong and a software engineer for SV Corporation pty. ltd., Sydney, Australia.

Farzad Safaei (farzad@uow.edu.au) is the professor of telecommunications engineering and director of Centre for Emerging Networks and Applications at the University of Wollongong, Australia. He is also the Program Manager of Smart Internet Technology Cooperative Research Centre, Australia.

Paul Boustead (boustead@titr.uow.edu.au) is a Senior Researcher in telecommunications engineering at the University of Wollongong, Australia. He is also the Chief Technology Officer for SV Corporation pty. ltd., Sydney, Australia.

Acknowledgments:

This work is supported by the Telecommunications and Information Technology Research institute of the University of Wollongong and the Smart Internet Technology Cooperative Research Centre, Australia.

Word count

2620 words without the title, text in the figures/table, captions, references, biographies and acknowledgments.

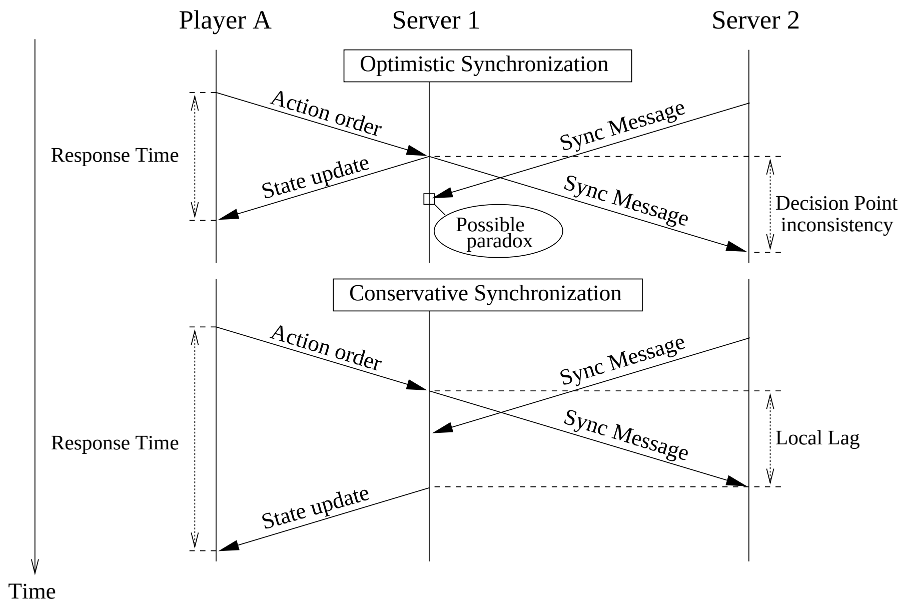

Fig. 1. Examples of synchronization in distributed server architecture. In optimistic synchronization, the sync message from server 2 can create a paradox on server 1 if it conflicts with Player A's action. The local lag compensates the inter-decision point inconsistencies assuming there is no jitter or packet loss.

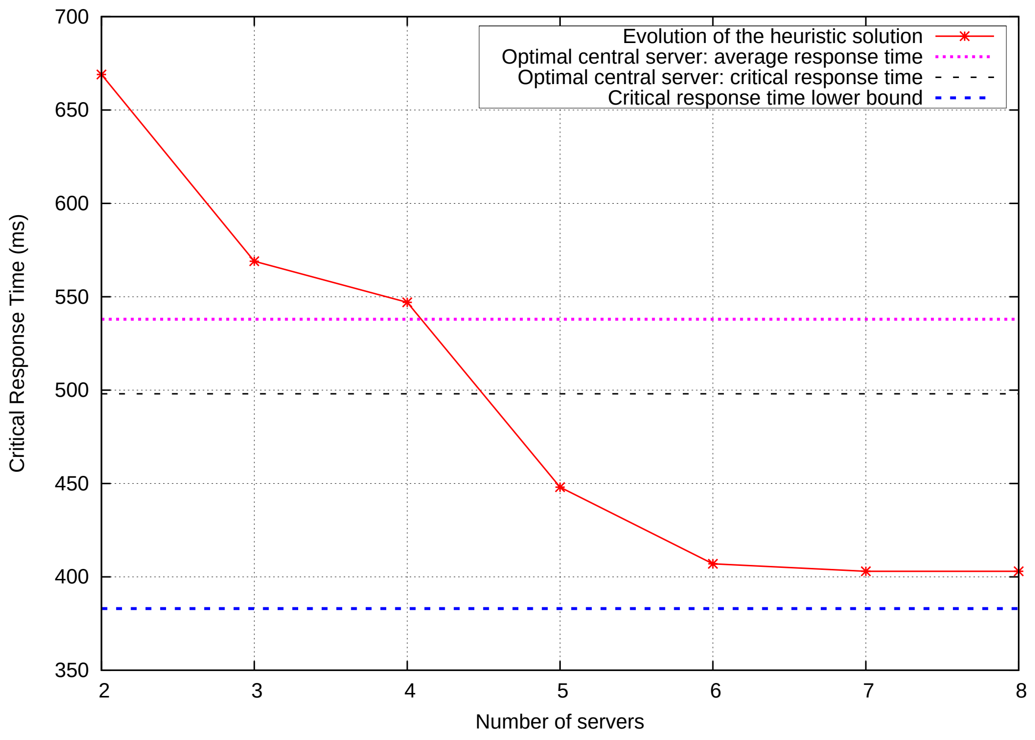

Fig. 2. Iterative evolution of a typical heuristic convergence. The critical response time is improved each time a server is added to the game. The final solution outperforms any central server approach and ends up close to the calculable lower bound.

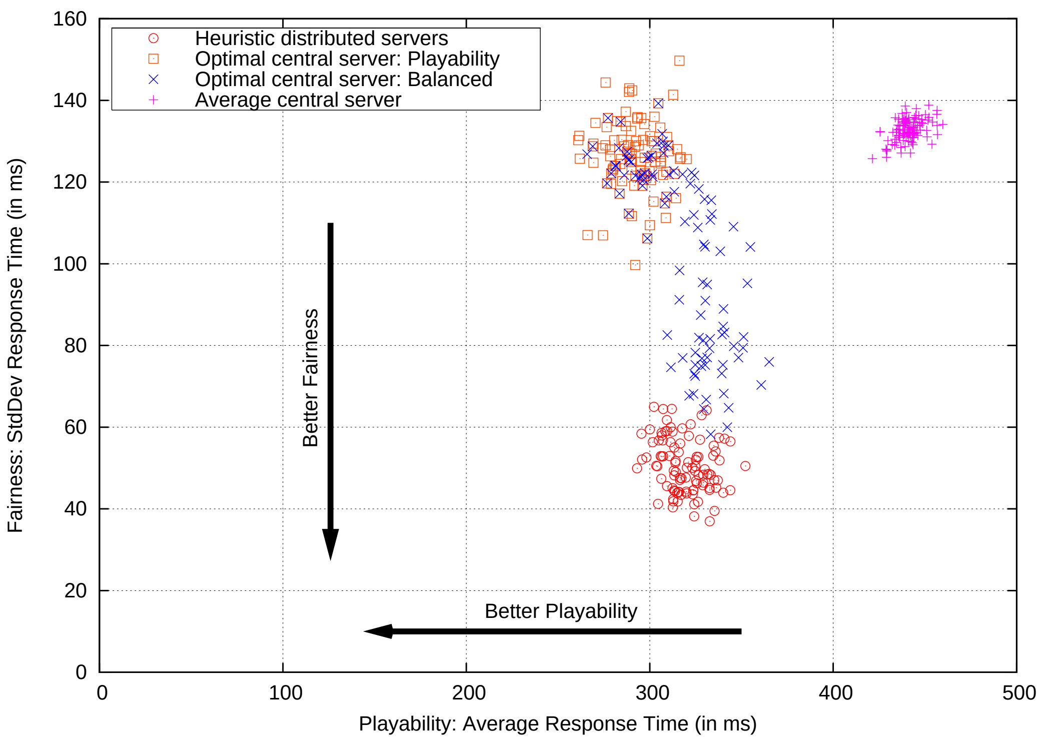

Fig. 3. Playability and Fairness of different decision point selection strategies. The set of distributed servers chosen by the heuristic outperforms the best central server approaches in terms of fairness at a marginal cost in average response time.