7-8-2006

On Latency and Player Actions in Online Games

Mark Claypool
Worcester Polytechnic Institute, claypool@wpi.edu

Kajal Claypool
kajal.claypool@gmail.com

On Latency and Player Actions in Online Games

Mark Claypool and Kajal Claypool
claypool@cs.wpi.edu kajal.claypool@gmail.com

July 8, 2006

1 Introduction

The growth and penetration of broadband access networks to the home has fueled the growth of online games played over the Internet. As we write this article, it is 5am on a typical weekday morning and Gamespy Arcade[1] reports more than 250,000 players online playing about 75,000 games! This proliferation of online games has been matched by an equivalent growth in both the variety of online games and in the support provided for online players. The spectrum of online games has shifted from a few people collaborating or competing on a Local Area Network (LAN) in first person perspective games such as id's *Doom* in the early 1990s, to thousands of players interacting over the Internet in a wide variety of games ranging from first person shooter games and role playing games to strategy and sports games. This escalation in the popularity of online games is also reflected in the correspondingly high number of servers spread across the globe that support and host thousands of players playing these games.

The best-effort nature of the Internet provides challenges for the real-time interaction required for online computer games, since there are no guarantees of network capacity, timely delivery or even delivery at all as Internet packets can be lost. Fortunately, by design, most online games have low bitrate requirements, sending frequent but small packets that are typically well-below the capacities of broadband or even dialup modem connections. In addition, the effects of packet loss can be mitigated by frequent game state updates or even by repair techniques. This leaves delayed delivery of packets, often in the form of network latency from a game player to a game server or other players, as the primary bottleneck for online game performance.

Previous studies have empirically determined the lower bounds for network latency for different types of networks. Typical LAN latencies, for example, are quite small, usually well under 10 milliseconds. Even Wireless LANs, increasingly popular for end-user connections, usually have latencies less than 10 milliseconds. For home users, the bulk of the game player population, latencies often depend upon the "last-mile" access networks. Dialup modems, for example, can add 100s of milliseconds of latency, while broadband access networks such as cable and asymmetric digital subscriber lines (ADSL) typically have lower latencies [8]. Cable modem latency, however, can vary considerably with worst case latencies of over 100 milliseconds. Once on the Internet, there are

[1]Gamespy provides popular services for the online gaming community, including game server browsing and player forums. Gamespy Arcade is online at http://www.gamespyarcade.com/.

lower bounds of approximately 50 milliseconds across a continent, with even higher latencies to cross over to other continents. Overall latencies can vary from 100s of milliseconds to 500 milliseconds up over even 1 second for some Internet connections [7].

Propitiously, not all aspects of player interactions are sensitive to latency. In particular, online games go through phases, where a game server is setup, players seek other players out, data is exchanged between game clients and local game data is loaded from the disk. None of these phases are sensitive to latency. Upon completion of the above phases, the online game proceeds to the play phase where players actually interact with the game world.

The play phase, arguably the most important and interesting aspect of online games, requires different types and levels of interactions between the player and the game. A first person shooter requires quick hand-eye coordination in moving the cross-hairs of a gun to target an opponent, a real-time strategy game requires thoughtful, but rapid, selection of units and buildings to create an army, while a sports game requires fluid key-presses and joystick movement to move an avatar in response to action on the screen. Even within a single genre, not all games are the same. For example, one first person shooter may have intense one-on-one combat with high precision weapons, while another may require strategic movement of teams of players and less frequent combat with lower precision weapons or even vehicles. The player actions in the play phase can be significantly impaired by latency.

However, not all player actions are equally tolerant to latency. Some actions such as shooting a sniper rifle at a moving opponent are greatly impacted by latency, while other actions such as selecting a set of troops and moving them across a battlefield tend to be less sensitive to latency. To explain this, this work contributes a novel categorization of the effects of latency on different player actions based on two salient action properties: the *precision* required to complete the action and the *deadline* by which the action must be completed. Actions with higher precision and tight deadlines are sensitive to even modest latencies, while actions with lower precision and loose deadlines are nearly impervious to typical Internet latencies. By this categorization, the effects of latency on sniping in a first person shooter (tight and precise) and troop selection in a real-time strategy game (loose and imprecise) can be explained in terms of their precision and deadline requirements. The categorization of actions is related to games in general, as well as popular game genres, through a new classification of games introduced in this article that emphasizes the player interaction model and the player perspective.

Thus, the goal of this work is to clarify the effects of Internet latency on online games by carefully examining the actions in online games studied thus far in the context of the proposed categorization of actions. This both validates the categorization of player actions presented here, and provides a framework for studying and engineering online games of the future. The results presented in this article are useful for: 1) game designers, so they may know the latency tolerances of different player actions in order to apply latency compensation techniques, as needed; 2) network designers, to create infrastructures to provide Quality of Service (QoS) for online games and other interactive applications; and 3) game players themselves, allowing them to make informed choices about their Internet connections or QoS purchases that may affect latency and hence game play.

Figure 1: *Doom 3*, An Avatar Game with a First Person Perspective.

Figure 2: *Madden NFL*, An Avatar Game with a Third Person Perspective.

2 Game Classification

A common conception among game players is that network latencies below 100 milliseconds are required for unimpaired game play, with maximum tolerable latencies being just over 100 milliseconds [4], regardless of the game genre. However, as not all games have the same interactions, it follows that latency does not effect all games equally. Games, and game genres, are typically defined by how the player interacts with the game world and by how the player views the game world on a screen. These two factors, *interaction model* and *perspective*, provide a game classification that helps determine the impact of latency on games.

Our game classification, based on [11], broadly organizes games into either the *Avatar* model or the *Omnipresent* model. In the Avatar model, the player interacts with the game through a single representative character and the player actions are defined in terms of commanding the character. The player's character, called the *avatar*, exists at a particular location in the virtual world and can influence only the immediate locality. Games with the Avatar interaction model typically have either a *first person* perspective where the player looks through the eyes of the avatar, or a *third person* perspective where the player follows an avatar in the virtual world. First person shooter (FPS) games, role-playing games (RPGs), action games, sports games and racing games are all examples of game genres that have an Avatar interaction model. These game genres often differ in the perspective, for example FPS games have a first person perspective while RPGs typically have a third person perspective. Some genres such as racing games allow the player to switch between a first person and third person perspective. For reference, Figure 1 is a screen shot of *Doom 3* showing an Avatar interaction model with a first person perspective, while Figure 2 is a screen shot of *Madden NFL* showing a third person perspective.

In the Omnipresent model, the player has the ability to view and influence simultaneously different aspects of the game world. While the player can not view or control the entire game world, the player is said to be *omnipresent* controlling the entire set of resources under the player's control. The player's actions, thus have a more global influence than do actions in an Avatar model. The perspective of games with the Omnipresent interaction model is often variable, giving players an aerial perspective to provide a bird's eye view of the virtual world, but also allowing players to zoom

Figure 3: *Warcraft III*, An Omnipresent Game, Featuring Real-Time Strategy Resource Management.

Figure 4: *Simcity 4*, An Omnipresent Game Centered on Construction and Simulation.

in to a third person perspective to provide finer granularity of control over individual resources. Real-time strategy games (RTS), and construction and simulation games (CMS) are examples of game genres with the Omnipresent interaction model. Figure 3 is a screen shot of *Warcraft III* showing the Omnipresent interaction model with an aerial perspective, while Figure 4 shows a screenshot of *Simcity 4*, also with an interaction model and an aerial perspective.

3 Game Phases

Online games go through phases that differ in the player's interactions with the game and in the network traffic generated. Although the duration and frequency of each phase varies depending upon the specific game, fundamental phases common to most online games include:

- *Setup* – During the Setup phase, players hosting a game wait for players to join the game, and all game players select starting parameters appropriate for the game they are playing. For instance, in a real-time strategy game, the hosting player would select the map and the starting resources, while the joining players would choose colors and teams. In a football game, the hosting player would choose the stadium and weather conditions while the joining player would choose a specific team and uniforms. Some games may have multiple Setup phases, such as a basketball tournament or a tennis circuit, while others have only one Setup phase, such as a dungeon crawl. The Setup phase typically has infrequent interactions between players since each player interacts with the local game only until the few setting choices are made. Thus, the Setup phase is marked by minimal network traffic and is not significantly affected by latency.
- *Synchronization* – After the Setup phase but before the gameplay actually begins, many games Synchronize game state and parameter settings between games. For example, a custom map or stadium selected may be sent from the game host to the other games, or the team selections and uniforms may be exchanged among games. The Synchronization phase is

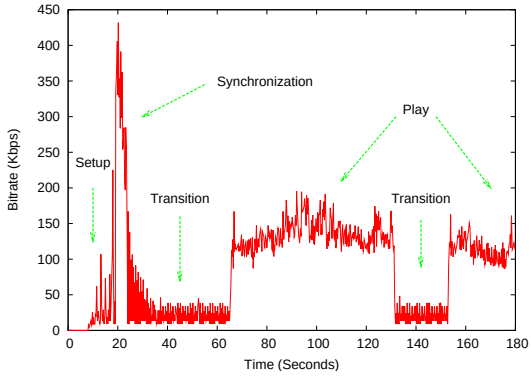


Figure 5: Example of Game Phases (*Untold Legends*)

Figure 6: Screenshot of *Untold Legends*, an Avatar Model, Third Person Perspective Game.

generally marked by high bitrates in order to exchange data as fast as possible to proceed on to gameplay. Players do not interact at all during Synchronization and so are unaffected by latency.

- *Play* – During the Play phase, the game is actually played, with players responding to the game state and their interactions communicated to other players, as appropriate. For example, a combat game might communicate the movement of an avatar and firing of a weapon to other players, while a hockey game may communicate the direction and velocity of the puck. The Play phase generally has moderate bitrates with frequent exchanges of small network packets to keep latency low, but without much data to exchange. It is during the Play phase that the effects of latency on player actions are of most interest and it is the core subject of this article.

- *Transition* – In between Play phases, some games have a Transition phase where game information is loaded and processed locally from a game disk into memory. For example, in an exploration game, the map may be loaded and the location of the puzzles and prizes determined, while in a racing game, the attributes of each car could be loaded and processed. The Transition phase generally has low network bitrates since most data is processed locally from disk and not over the network. Players do not interact during the Transition phase and so are unaffected by network latency.

Figure 5 depicts an example of the game phases for the Sony PSP game *Untold Legends*, a third person action game where players do a dungeon crawl (a screenshot is shown in Figure 6). During the Setup phase, players load avatars used in previous games or create new avatars and the hosting player decides where in the story to start. During the Synchronization phase, game state information is communicated between games, such as the quests that have been completed and magic items that have been found. During the Transition phases, the players choose to move between different world locales (such as from a town into the woods or from the woods into a dungeon) and local game data is loaded from the disk and processed in memory. During the Play phases, the players interact with the game, mostly by controlling their avatars through movement, combat and inventory management.

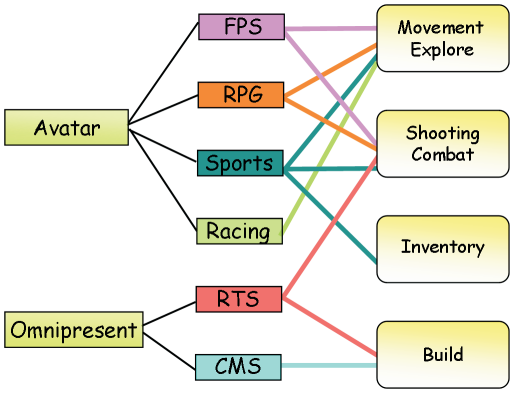


Figure 7: Primary Player Actions during the Play Phase of Different Game Genres.

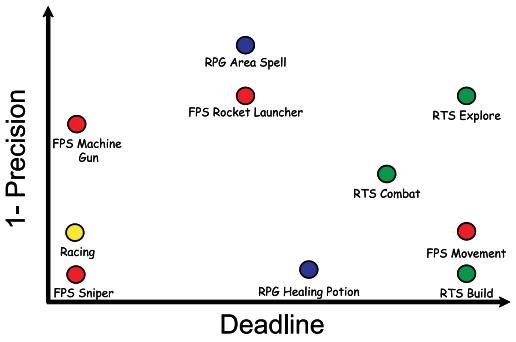


Figure 8: Taxonomy of Different Player Actions along the Precision and Deadline Axes.

In general, the length and frequency of each phase depends upon the game and often on the player choices made within the game. For example, the length of a half in a sports game or the frequency of a player returning to town for healing in a combat game directly determine the length of a play phase.

4 Player Actions

The Play phase of a game can be further categorized by the different types of player actions. For example, in first person shooter (FPS)[2] games the two most common types of player actions are *movement* and *shooting*, with movement shifting the view of the player in the virtual world, and shooting placing the cross-hairs of a weapon on the target and firing. Similarly, during the Play phase in a real time strategy (RTS) game, player actions can be classified in terms of *build*, *combat* and *explore*. Build begins construction of a building, such as barracks to recruit soldiers, combat instructs avatars to engage in battle and explore moves avatars (as opposed to movement of the player perspective in an FPS) in the virtual world. The Play phase of other game genres can be similarly classified in terms of the player actions. Figure 7 summarizes the primary player actions for the Play phase of different game genres (FPS, RTS, Sports, and Racing). It should be noted that individual games may vary in the quantities of each type of interaction. For example, some FPS games may require lots of shooting with little movement while others may require more movement and less shooting.

Across all genres, player actions vary along two primary axes, *deadline* and *precision*. Deadline is the time required to complete the action, that is the length of time it takes to achieve the final outcome of the action. For example, in Diablo 2 deadline for a portal is the time it takes for an avatar to read a magic scroll and invoke a town portal that will transport the avatar to town. Precision is the degree of accuracy required to complete the interaction successfully. For example, in Battlefield 1942 precision is the accuracy required to shoot a distant enemy with a sniper rifle.

[2]For better exposition, this section uses popular game genres rather than the perspective introduced in Section 2. This can be directly translated to perspectives, as appropriate.

Different player actions have disparate deadline and precision requirements. This disparity can even be observed across game genres, within a game genre and even within a specific game. For example, shooting in an FPS game generally has high precision and tight deadline requirements, meaning the player must place the gun cross-hairs exactly on the target to hit and the action must be carried out immediately or the target may move. However, the precision and deadline requirements for shooting can vary with the weapon used. For example, shooting with a sniper gun requires high precision with a tight deadline, shooting with a machine gun relaxes both the precision and deadline, and shooting with a rocket launcher imposes relatively lower precision and deadline requirements than either the sniper gun or the machine gun.

Movement in an FPS game requires high precision but has a relatively looser deadline requirement than does shooting, implying the precise location will determine if a player's avatar can be hit, while moving from one location to another takes on the order of seconds. Movement in FPS games and exploration in RTS games are similar user actions, but have different precision, with RTS exploration generally having lower precision than FPS movement. RTS exploration often moves a large number of troops towards an area in the virtual world, as opposed to an exact location for an FPS movement. However, the two interactions often have comparable deadline requirements since moving across the virtual world can take a similar amount of time.

Figure 8 shows a taxonomy of the different player interactions along the precision and deadline axes. The x-axis is the deadline requirement and the y-axis is amount of imprecision (indicated as 1 - Precision). The FPS Sniper has high precision and a tight deadline, RTS Build has a high precision but a loose deadline, RTS Combat has a lower precision than either FPS Sniper and RTS Build, but a looser deadline than FPS Sniper and a tighter deadline than RTS Build.

In general, the further away an action is from the origin in the Precision-Deadline plane, the less the impact that latency has on player performance. Thus, FPS Sniper and Racing are sensitive to latency, while RPG Area Spell and RTS Explore are less sensitive to latency.

5 Player Actions and Latency

Most games today run on a client-server architecture with a single, authoritative server that handles the game logic. When a player performs an action, the client sends a message to the server. The server processes the action and sends any changes to the game state back to the waiting client to render on the local display. The client then renders the new game state to the player and the process repeats. Note, in cases where a player "hosts" a game, that player's computer then acts like a server for all players, as well as a client for the local player. This is still fundamentally a client-server architecture, even though it may look like client-client (or peer-to-peer) on the surface.

All player actions in the client-server architecture are delayed by the round-trip latency between client and server. If the latency between the client and server is large enough, the player is aware of the delay between the commands given to the game and the response of the game. This delay, or latency, can degrade online game performance. With online games played over the Internet, latency for an action can be attributed to many network components, such as the time to transmit the encoded action in an IP packet, the time for the packet to propagate from one link to another, and time spent waiting in a router queue during network congestion.

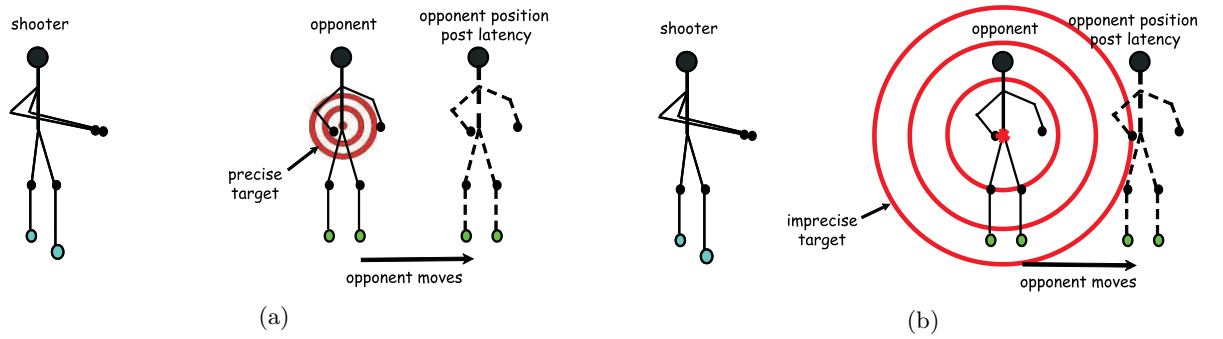


Figure 9: (a) Targeting Opponent with High Precision Weapon. (b) Targeting Opponent with Low Precision Weapon.

The deadline and precision requirements for a given player action determine the effects of latency on that action.

Precision. Consider a shooting action where the player targets an opponent moving across the field of view from left to right, depicted in Figure 9. With a high precision weapon (see Figure 9(a)), for example a sniper rifle, the player on the left sees the opponent as the solid outline, with the target circle representing the precision of the sniper gun the player is shooting. When the player aims and shoots, the gun will hit any opponent within the circle. However, with latency between the player action and the game server recording that action, the opponent is no longer at the solid outline, but instead has moved to the right to the dashed outline, resulting in a miss. However, when the player is shooting a weapon with lower precision (see Figure 9(b)), such as a machine gun, the target circle is larger. In this case, the latency between the player action and the game recording that action still allows the opponent to move, but the opponent remains within the target area, enabling the player to score a hit.

This example illustrates our first insight: *For a given game action, the higher the precision required the greater the impact of latency on performance.*

Deadline. Consider a real-time strategy game where a player must construct a factory to produce goods, as depicted in Figure 10. The player selects a location and instructs the construction to begin. When the deadline to complete the building is tight as in Figure 10(a), a small amount of additional latency is relatively large and causes the building to take much longer to complete relative to the build time without latency. However, when the deadline to complete the building is loose as in Figure 10(b), the same amount of latency is no longer as significant, and the building takes approximately the same amount of time to complete.

This example illustrates our second insight: *For a given game action, the tighter the deadline the greater the impact of latency on performance.*

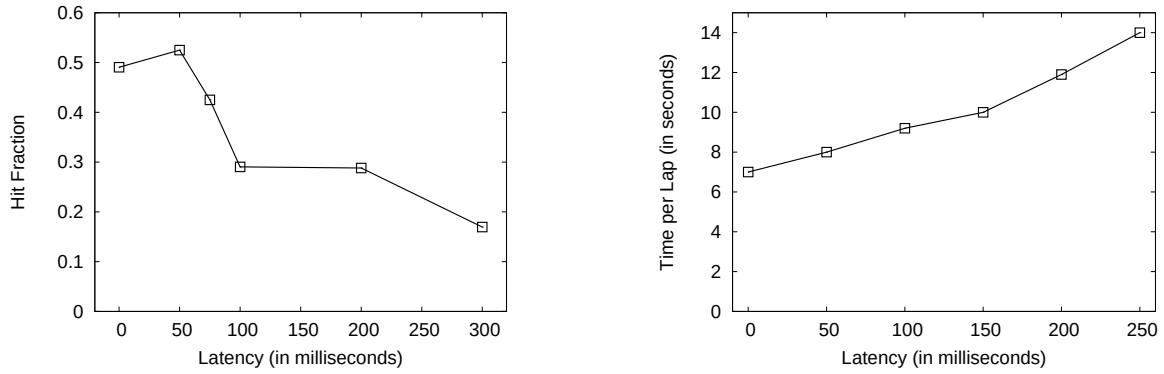
There have been various studies that have measured the impact of latency on performance for different player actions in several different games [6, 1, 2, 3, 4, 5, 9, 10]. These studies have generally setup a network testbed that allows for careful control of network latency, typically by having the online game played on a Local Area Network (LAN) and adding a middle-ware router and application to add a controlled delay to all network traffic. While these studies were not setup

(a) (b)

Figure 10: **(a)** Constructing a Building with a Short Deadline. **(b)** Constructing a Building with a Long Deadline.

specifically informed by the insights provided by this article, they allow illustration of the effects of latency on performance for player actions with various deadline and precision requirements.

In all of the below graphs, the x-axis is the amount of latency in milliseconds induced in the experiments, while the y-axis is the measure of performance specific to the particular game. For some graphs, a higher number on the y-axis is better, such as the accuracy of shooting a weapon. For other graphs, a lower number is better, such as the time to move from point A to point B.

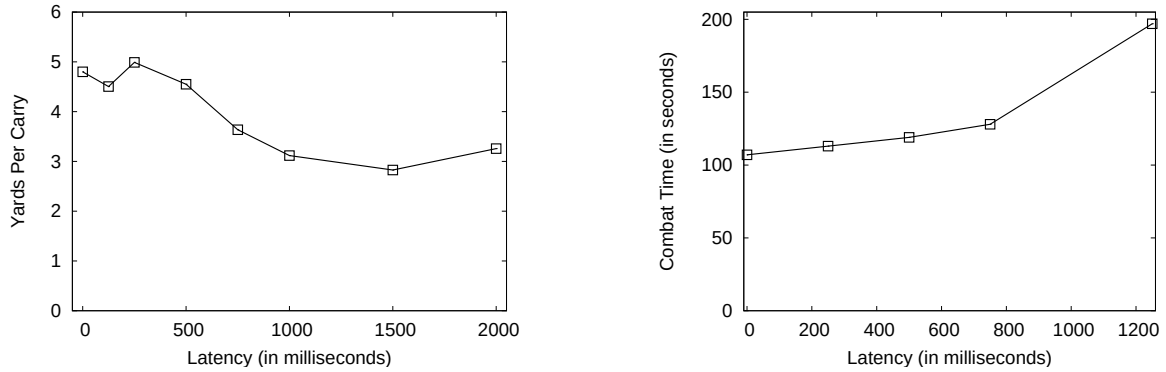


(a) (b)

Figure 11: Avatar Model, First Person Perspective. **(a)** Hit Fraction versus Latency (*Unreal Tournament 2003*, a First Person Shooter Game). **(b)** Lap time versus Latency (*RC Racing*, a Racing Game).

Figures 11(a) and 11(b) show the effects of latency for two games in the Avatar model with a first person perspective. Figure 11(a) depicts the effects of latency on shooting a high-precision gun in *Unreal Tournament 2003*, a first person shooter [1]. The experiments measured the average hit fraction during two-player battles with high precision weapons. There is a noticeable overall downward trend in performance as latency increases, with a sharp drop (about 35%) in accuracy

at 100 milliseconds of latency. Figure 11(b) depicts the effects of latency in a car racing game [10]. The experiments measured the time to complete a lap around a race track for subjects of varying degrees of driving skill. There is a noticeable upward trend in lap time as latency increases, with a significantly steeper increase above 50 milliseconds and again above 150 milliseconds.



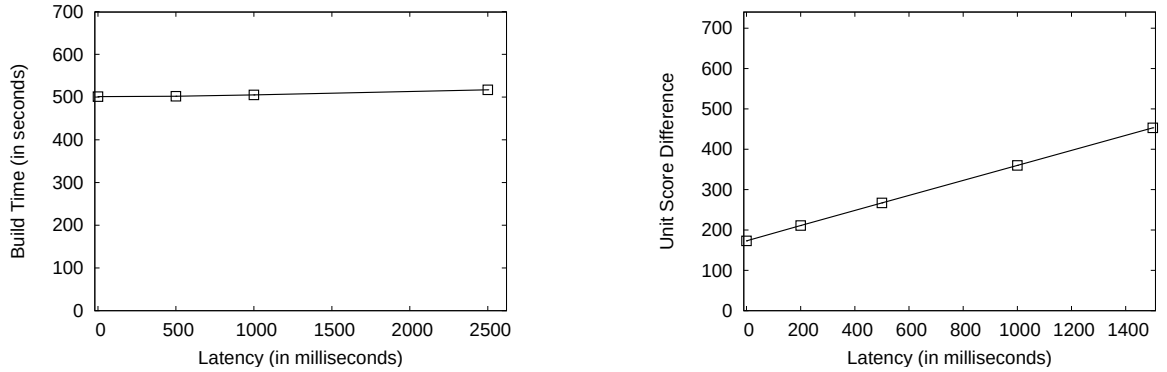
(a) (b)

Figure 12: Avatar Model, Third Person Perspective. **(a)** Yards per Attempt versus Latency (*Madden NFL football*, a Sports Game). **(b)** Combat Length versus Latency (*Everquest 2*, a Third Person Role Playing Game).

Figures 12(a) and 12(b) show the effects of latency for two games in the Avatar model with a third person perspective. Figure 12(a) depicts the effects of latency on running[3] in *Madden NFL*, a sports game [9]. The experiments had players attempt a running play each time, and measured the average yards per attempt. While there is an overall downward trend in the effects of latency, there is no significant drop-off in performance until after 500 milliseconds of latency. Figure 12(b) depicts the effects of latency on the length of combat in *Everquest 2*, a third person action game [5]. These experiments had clients connect to an actual Everquest server over the Internet, while inducing a controlled amount of additional latency near the client. With an increase in latency, the avatar's ability to deal damage decreases, making it take longer to complete a fight. However, the relative decrease in performance from 0 to 500 milliseconds is small, adding only an additional 5 seconds to about 2 minutes of combat.

Figures 13(a) and 13(b) show the effects of latency for two games with an Omnipresent model. Figure 13(a) depicts the effects of latency on the time to construct the technology tree for Humans in *Warcraft III*, a real-time strategy (RTS) game [2]. The experiments measured the build time versus latency for all experimental runs, as well as a best-fit line for the data. When there is no induced latency, building the technology tree takes about 8 minutes, while latency values of up to 3.5 seconds increase total build time by at most 14 seconds, which is less than 1% of the total time required to complete this action. Figure 13(b) depicts the effects of latency on the unit score difference from combat between two armies in *Age of Mythology*, also a real-time strategy game. The experiments pitted players with two small, equally matched armies against each other. The unit score difference is the player without latency's unit score minus the player with latency's unit score. There is a slight upward trend in that the score difference increases as latency increases, but

[3]The avatar carries the football and runs as far as possible without getting tackled by the opposing team.



(a) (b)

Figure 13: Omnipresent Model. **(a)** Build Time versus Latency (*Warcraft III*, a Real-Time Strategy Game). **(b)** Unit Score Difference versus Latency (*Age of Mythology*, a Real-Time Strategy Game.

the visual correlation is slight. Moreover, the difference in score from no induced latency to one second of induced latency is only equivalent to about one unit, an insignificant amount in the large battles that are typical in most RTS games.

6 Summary

Figure 14 summarizes performance degradation for different classes of online games, depicted by an exponential curve fit to the measured data. Online games that use the Avatar model of player interaction are more sensitive to latency than games that use the Omnipresent model. Further within the Avatar model, games that use the first person perspective are more sensitive to latency than games that use the third person perspective. Within a given game, player actions that are less precise or have looser deadlines tend to shift the curves in Figure 14 to the right, while more precision and tighter deadlines shift the curves left. Within a given class of games, the relative amounts of different player actions determine the exact location of the curve. For example, an FPS game with more movement and less precise shooting may have the blue curve in Figure 14 shifted more to the right and flattened.

The horizontal gray area in Figure 14 is a visual indicator of player tolerances for latency. Although the exact threshold depends upon the game and to some extent the player, generally performance degradations above this threshold are acceptable while performance degradations below this threshold are unacceptable. Table 1 summarizes this effect of latency on performance in online games.

Model	Perspective	Example Genres	Sensitivity	Thresholds
Avatar	First Person	FPS, Racing	High	100 milliseconds
	Third Person	Sports, RPG	Medium	500 milliseconds
Omnipresent	Varies	RTS, Sim	Low	1000 milliseconds

Table 1: Summary of Latency and Online Games

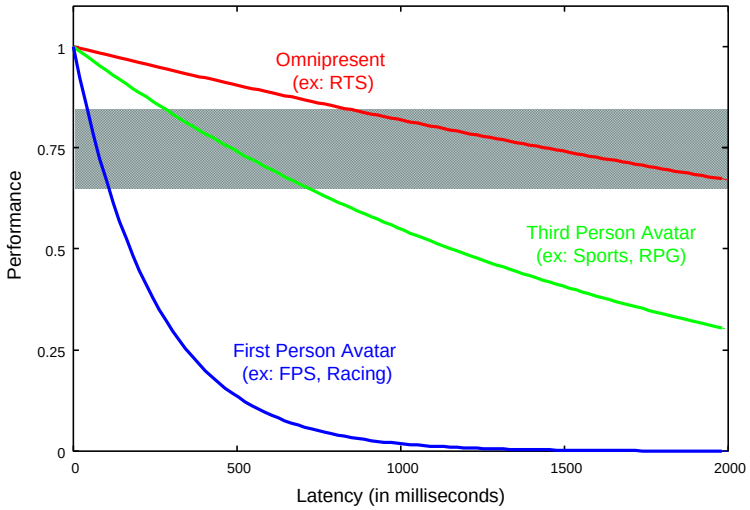


Figure 14: User Performance Under Different Induced Latencies for Several Classes of Games. Above the grey region, quality is generally acceptable while below the gray region, quality is generally unnacceptable.

References

[1] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games Workshop (NetGames)*, September 2004.

[2] Mark Claypool. The Effect of Latency on User Performance in Real-Time Strategy Games. *Elsevier Computer Networks, Special Issue on Networking Issues in Entertainment Computing*, 49(1):52–70, September 2005.

[3] Mark Claypool, Kajal Claypool, and Feissal Damaa. The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games. In *Proceedings ACM/SPIE Multimedia Computing and Networking (MMCN) Conference*, San Jose, CA, USA, January 2006.

[4] Matthias Dick, Oliver Wellnitz, and Lars Wolf. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of the 4th ACM Network and System Support for Games (NetGames)*, October 2005.

[5] Tobias Fritsch, Hartmut Ritter, and Jochen H. Schiller. The Effect of Latency and Network Limitations on MMORPGs: a Field Study of Everquest 2. In *Proceedings of the 4th ACM Network and System Support for Games (NetGames)*, October 2005.

[6] Tristan Henderson and Saleem Bhatti. Networked Games - a QoS-sensitive Application for QoS-insensitive users? In *Proceedings of the ACM SIGCOMM Workshop on Revisiting IP QoS*, pages 141 – 147, August 2003.

[7] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, Jim Kurose, and Don Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE Infocom*, Hong Kong, China, April 2004.

[8] Tom Jehaes, Danny De Vleeschauwer, Toon Coppens, Bart Van Doorselaer, Eva Deckers, W. Naudts, K. Spruyt, and R. Smets. Access Network Delay in Networked Games. In *Proceedings of the ACM NetGames Workshop*, May 2003.

[9] James Nichols and Mark Claypool. The Effects of Latency on Online Madden NFL Football. In *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2004.

[10] Lothar Pantel and Lars C. Wolf. On the Impact of Delay on Real-Time Multiplayer Games. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[11] Andrew Rollings and Ernest Adams. *On Game Design*. New Riders, 2003. ISBN: 1-5927-3001-9.