Latency Can Kill: Precision and Deadline in Online Games

Mark Claypool
Interactive Media and Game Development
Worcester Polytechnic Institute
Worcester, MA, 01609, USA
claypool@cs.wpi.edu

Kajal Claypool
Weather Sensing Group
MIT Lincoln Laboratory
Lexington, MA, 02420, USA
claypool@ll.mit.edu

ABSTRACT

The growth in network capacities and availability has been accompanied by a proliferation of online games. While online games perform well under many network conditions, Internet delays can often degrade online game performance. The precise impact that latency has on online gameplay depends upon the game type and the actions within the game. While the effects of latency on specific games has been studied, knowledge about the effects of latency on classes of games and about the effects of latency on different player actions is lacking. This paper presents a broad, perspective-based classification of games based on player control and camera view. The foundation of the game classification are player actions, each of which is defined by its precision and deadline requirements. Experiments with controlled amount of precision, deadline and latency support the classification. This classification of games should prove useful for game designers, network engineers and game players themselves as they build and play on tomorrow's networks.

Categories and Subject Descriptors

J.m [**Computer Applications**]: Miscellaneous

General Terms

Latency, Games, Performance

1. INTRODUCTION

The growth in availability and capacity of access networks to the home has spurred the development of online games. Correspondingly, online games have seen an equivalent growth in the variety of game choices that are available for online players. While early online games featured a few people collaborating or competing on a LAN in first person perspective games such as id's *Doom* in the 1990s, today thousands of players interact over the Internet in a wide variety of games ranging from first person shooter games and role playing games to strategy and sports games. The escalation in the popularity of online games has been supported by a high number of geographically dispersed servers that host the thousands of online game players.

The best-effort nature of the Internet poses several challenges for the real-time interaction required for online games – there are no guarantees of network capacity, timely delivery or even delivery at all as Internet packets can be lost. Most online games are designed to have low bitrate requirements, frequently sending small packets at rates well below the capacities typical of broadband or even dialup modem connections. The effects of packet loss are often mitigated by updating the game state frequently or by employing packet repair techniques. The lack of timely delivery of packets, manifesting as delayed information from a game player to a game server or other players, remains 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 only a few milliseconds. Even Wireless LANs, increasingly common as an end-host connection, usually have latencies less than 10 milliseconds. For online game players at home, 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 modems and DSL typically have lower latencies. Still, cable modem latency, for example, can have worst case latencies over 100 milliseconds. Once past the access link, there is roughly a lower bound of approximately 50 milliseconds of latency to cross a continent, with even higher latencies to other continents. Overall latencies can, thus, vary from 10s of milliseconds into the 100s of milliseconds and even 1 second for some Internet connections [6]. This delay in the arrival of packets can severely degrade the performance of online games and ruin the fun for the players.

Fortunately, not all forms of player-to-game interactions are sensitive to latency. In particular, online games go through phases, where a game host 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. However, upon completion of the above phases, the online game proceeds to the play phase where players actually interact with the game world and are vulnerable to network latencies.

The play phase, arguably the most important and interesting aspect of online games, has different types of interactions between the player and the game. A first person shooter requires quick hand-eye coordination in moving the cross-hairs of a weapon to target an opponent, a real-time strategy game requires thoughtful, but rapid, selection of units and buildings to muster an army, while a sports game requires fluid key-presses and joystick manipulation 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.

MMSys'10, February 22–23, 2010, Phoenix, Arizona, USA.

The exact effects of Internet latencies on online games depends upon an understanding of the interactions required within a specific game. Not all player actions are equally tolerant to latency. Some actions such as shooting a sniper rifle at a moving target 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 these differences, our position paper [3] proposed a novel categorization of the effects of latency on different player actions based on two properties core to each player action: the *precision* required to complete the action and the *deadline* by which the action must be completed. Actions with high precision and tight deadlines are sensitive to even moderate latencies, while actions with low precision and loose deadlines are resilient to even high 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, including popular game genres, through a new classification of games introduced to emphasize the player interaction model and the player perspective.

This paper describes game genres and player actions and revisits the proposed precision-deadline model to provide a clear point of reference through which to view player actions. The paper also provides experimental data to support the proposed precision-deadline model. To provide this support, a publicly available game, *BZFlag*,[1] was modified to allow performance measurements of actual online games with controlled, known amounts of latency. While BZFlag has only two primary forms of player interaction, notably driving a tank and shooting at an opponent, different amounts of precision and deadline were emulated by changing the tank size (affecting precision) and bullet speed (affecting deadline). Hours of experiments varying precision and deadline over a range of latencies provides quantifiable data to support the precision-deadline model.

The results presented in this paper generally support the idea that game actions are affected differently by latency depending upon their precision and deadline requirements. Games that had higher precision requirements (through smaller tanks) and games that had tighter deadline requirements (through faster bullets) were generally more affected by latencies than games with lower precision and looser deadlines. However, it should be noted that the variance in player performance can, at times, be more pronounced than the effects of latency itself.

This paper clarifies 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. The results presented in the paper validate the categorization of player actions, and the precision-deadline model provides a framework for studying and engineering online games of the future. The results presented in this paper 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 that provide Quality of Service (QoS) for online games and other interactive applications; and 3) game players themselves, enabling informed choices about their Internet connections or QoS purchases that may affect latency and hence game play.

The rest of this article is organized as follows: Section 2 presents our game classification based on player perspective; Section 3 describes the phases common in most online games, with the goal of concentrating on the interactive play phase; Section 4 details our taxonomy of player actions along the precision and deadline axes; Section 5 provides insights into how latency may affect player actions with different precision and deadline requirements; Section 6 describes our approach to measure the effects of latency on actions with different precision and deadline requirements; Section 7 presents the results of the experiments; and Sectionconclusions highlights our conclusions and mentions some possible future work.

[1] http://BZFlag.org/

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

2. GAME CLASSIFICATION

A common conception among game players is that network latencies below 100 milliseconds are essential 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 and should not effect all games equally. Games, and game genres, are typically defined by how the player interacts with the game world (interaction) and by how the player views the game world on a screen (perspective). These two factors, *interaction model* and *perspective*, provide the basis for a game classification that helps determine the impact of latency on games.

Our game classification, based on [7], 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

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

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

and influence simultaneously different aspects of the game world. While the player can not always view or control the entire game world, the player is said to be *omnipresent* in that the player can control the entire set of resources under his/her control. The player's actions, thus, have a more global influence than 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 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 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 Omnipresent interaction model and an aerial perspective.

3. GAME PHASES

Online games go through phases that differ both in the player's interactions with the game and in the network traffic that is 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 other players to join the game, and all game players select

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

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 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 a selected stadium 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 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 is generally characterized by moderate bitrates with frequent exchanges of small network packets in an effort to keep latency low. 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 paper.

- *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 pro-

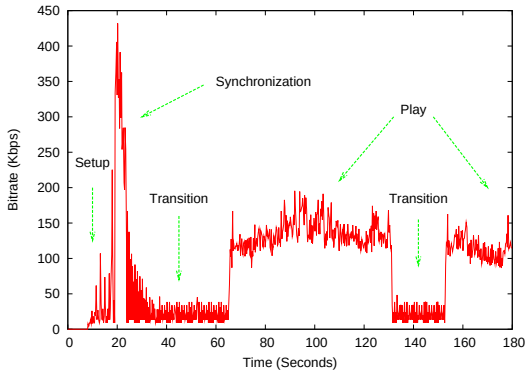

Figure 5: Example of Game Phases (*Untold Legends*).

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

cessed. 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 network traffic during 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 the starting point in the story. During the Synchronization phase, game state information is communicated between games, such as the quests that have been completed and the 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.

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 as *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 the player avatars (as opposed to movement of the player and hence their view 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 have lots of shooting with little movement while others may have 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 spell is the time it takes for an avatar to read a magic scroll and invoke a town portal that will transport the avatar back 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.

Different player actions have disparate deadline and precision requirements. This disparity can 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, implying that 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 is 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 player 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 the 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, a looser deadline than FPS Sniper but a tighter deadline than RTS Build.

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

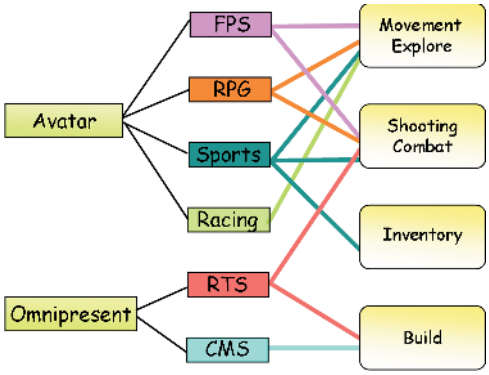

Figure 7: Primary Player Actions during the Play Phase of Different Game Genres.

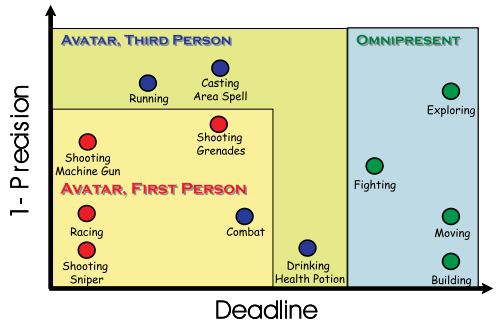

Figure 8: Taxonomy of Different Player Actions along the Precision and Deadline Axes.

In general, the further 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 modern-day games run on a client-server architecture typically 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, the host player's computer is a server for all players, as well as a client for the local host player. This is still fundamentally a client-server process architecture, in-spite of the client-client (or peer-to-peer) surface interaction that it might portray.

All player actions in this client-server architecture are delayed by the round-trip latency between the client and server. If the latency between the client and server is large enough, the player is not only aware of the delay between the commands given to the game (the player action) and the response of the game, but the delay, or latency, can also degrade online game performance. How much the player's action (and hence performance) is impacted by the latency is determined by the deadline and precision requirements of a given player 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 shotgun, 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.*

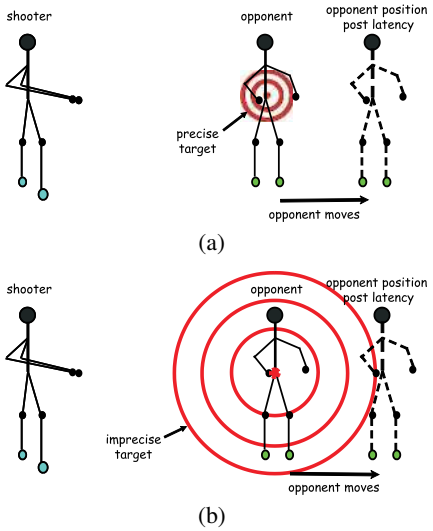

Figure 9: (a) Targeting Opponent with High Precision Weapon. (b) Targeting Opponent with Low Precision Weapon.

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.*

6. APPROACH

Our approach to evaluate and empirically validate our precision-deadline model (Section 4) and insights (Section 5) was to modify an open source, online game to allow for controlled precision-deadline experiments over a range of latencies.

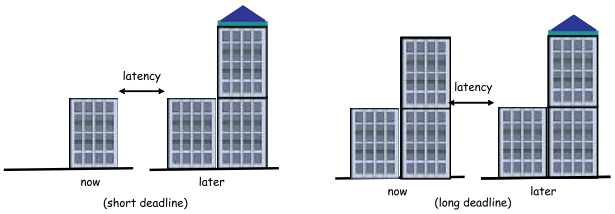

Figure 10: **(a)** Constructing a Building with a Tight Deadline. **(b)** Constructing a Building with a Loose Deadline.

Figure 11: *BZFlag*, A Third Person Avatar Game in which Players Drive a Tank and Shoot Opposing Tanks.

The game *BZFlag*,[3] a free online multiplayer 3D tank battle game, met our criteria for game selection. As per the genre classification in Section 2, BZFlag is a third person avatar game. Players command a single tank shown with a third person camera, driving around a fixed-size map and shooting opposing tanks, as depicted in Figure 11. Combat can occur between human-controlled tanks or between computer-controlled tanks (otherwise known as *bots*) or with some combination of both. Statistics regarding the number of servers and players currently playing are continuously being tracked online.[4] BZFlag is still popular in the online community, with an average of 20 servers with 2+ players when sampled during the late morning over several consecutive days.

6.1 Player Performance

The performance of a player is measured in terms of the score, calculated as the difference between the number of kills and the number of deaths. Computer players are all evenly matched with each other and generally accrue a score of zero under equal latency conditions. Thus, any observed differences in scores between players can be attributed to differences in latencies. The hit rate and the number of shots fired were recorded, as well. As a rough approximation of the amount of action, during a typical bot-vs-bot game on the default map, the kill and/or death rate is about 1.5 per minute with a hit rate of about 25%.

[3]The name *BZFlag* originates from "Battle Zone capture the Flag."
[4]http://stats.bzflag.org/

6.2 Modifications

The goals of our game modifications were 1) allow control of the precision and deadline of a tank's shooting actions, and 2) add the ability to induce fixed amounts of latency on a per-tank basis.

The modifications for controlling precision and deadline factors were fairly straightforward. Precision, the accuracy required for an action, was controlled by changing the tank size. Smaller tanks are a smaller target requiring more precision to hit, and vice versa for larger tanks. Deadline, the time required to complete an action, was controlled by changing the bullet speed. Faster bullets reached their targets more quickly and so had tighter deadlines, and vice versa for slower bullets. Changes to the code were verified visually and via log messages.

In the course of testing these modifications, it was observed that smaller tank sizes made the bots perform particularly poorly. In BZFlag, a tank may only fire one bullet at a time and must wait until the previous shot expires (about 3.5 seconds) before shooting again. By default, a bot shoots at an opposing tank if the miss will be less than one-half a tank length. After careful experiments, it was determined that a lower value of one-eighth the tank length produced better bot performance at all bullet speeds and reasonable performance even with small tank sizes.

The granularity of time keeping in BZFlag was insufficient for our measurement purposes. New time functionality that allowed millisecond precision for latency timing was added. However, since tank actions are updated only when the BZFlag game engine updates (about 60 times per second), the fidelity of induced latencies is only within about 15 milliseconds.

A new class called a `LatencyManager` was added to the code to act as a proxy between the bots and the server, delaying messages for a controlled amount of time. The original `ServerLink::send()` method in the code was modified to invoke the `LatencyManager` to delay packets, if necessary.

The initial intent had been to add latency for all packets sent to/from the server for a particular node. However, testing and code inspection determined that BZFlag clients determine whether or not bullets collide with their tanks. For our experiments, having collision detection in the client meant that opponents shots were lagged to the client, also, making a distant player more difficult to hit. Since this method leaves the game open for cheating by malicious players, most online games do collision detection in the trusted server. Since our intent was to explore the effects of latency for more typical game architectures, only shooting messages were lagged. Delaying only the shot messages allows non-distant clients to react to the position of a distant opponent naturally, but makes the distant opponent slow to react to closer players.

Tables 1 and 2 depict some of the results from pilot tests used to validate our code changes. Table 1 shows the result of a six-hour, 1v1 baseline test where neither player is lagged. After six hours, the scores and hit percentage are nearly the same. Table 2 shows the result of a six-hour, 1v1 baseline test where player 1 is lagged by 1000 milliseconds. After six hours, player 2 has a substantially higher score and better hit percentage.

6.3 Baseline

Our baseline had eight computer-controlled tanks (also known as *bots*), divided into four competing teams of two players each. Only these uniform, computer-controlled tanks were used, as opposed to human-controlled tanks, in order to decrease differences in performance that might be caused by differences in skill. Eight tanks provided enough action (shots and kills) in a short amount of time without having a map that was too crowded. Eight is also a common limit for online, FPS game tournaments.

Player	Hit%	Kills	Deaths	Score
1	24%	640	646	-6
2	25%	646	640	+6

Table 1: Six-Hour 1-vs-1 Control Test. Neither player is lagged.

Player	Hit%	Kills	Deaths	Score
1	26%	544	881	-337
2	32%	881	544	+337

Table 2: Six-Hour 1-vs-1 Latency Test. Player 1 is lagged.

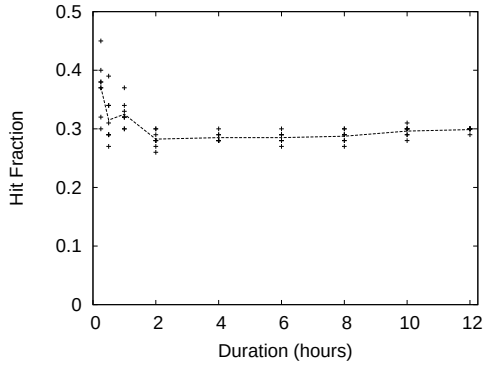

Figure 12: Hit Percentage versus Duration.

6.4 Duration

BZFlag can be played without an end-game condition, so a decision needed to be made as to how long to run each game. Short games are preferred, if possible, since that allows more tests to be run in a shorter amount of time. Long games, however, minimize any variance that might be caused by startup conditions.

Tests with the baseline game were run for durations of up to 12 hours, depicted in Figure 12. Each cluster represents an eight-player game run for the duration indicated on the x-axis and each dot shows the hit fraction on the y-axis for a single tank during that game. A line connects the average hit fraction for each cluster. From the figure, durations less than one hour have noticeably more variance across tanks than higher durations. Game durations of two-hours and above have little difference in the hit-fraction of the tanks. Thus, two-hours was chosen as the test duration for all subsequent tests.

6.5 Factors

Assuming BZFlag had been balanced for fair and fun gameplay in its latest incarnation, the experiments were designed to investigate the effects of more and less precision and tighter and looser deadlines on player performance with lag. Thus, the baseline values had the default precision and deadline with binary increments (both higher and lower) used for all factors.

For induced latency values, a latency of 0 milliseconds represents a client that is also acting as a server. A latency of 100 milliseconds is approximately the tolerable threshold measured for some first person shooter games [1]. A latency of 500 milliseconds provides a measurable degradation for third person avatar games [5]. A latency of 1000 milliseconds is considered quite high for most games and provides noticeable degradation to third person avatar games [5] and even to omnipresent games [2]. Table 3 presents the

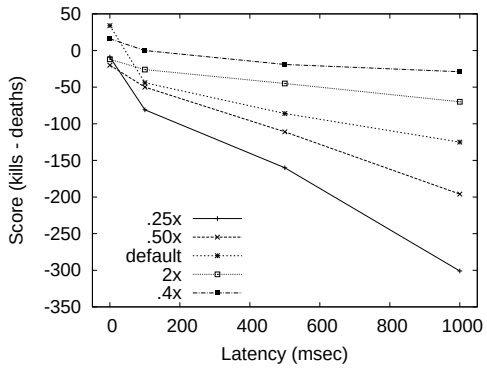

Figure 13: Score versus Latency for Different Precision Requirements (i.e. Tank Sizes).

factors used in the experiments.

Factor	Value
Tank Size	0.25x, 0.50x, default, 2x, 4x
Bullet Speed	0.25x, 0.50x, default, 2x, 4x
Latency	0 ms, 100ms, 500ms, 1000ms

Table 3: Experimental Factors and Values.

All games were run on machines that more than met the modest hardware requirements for BZFlag,[5] easily allowing 8 bots to play at once with no noticeable performance degradation.

7. RESULTS

Figure 13 shows the analysis of the data to evaluate latency and precision. The x-axis is the latency (in milliseconds) induced, and the y-axis is the score (kills-deaths). Each data point is the score achieved by one player (bot) with induced latency (the other players all had no latency). The trend lines are for games with different tank sizes (tanks for all players had the size indicated). With no induced latency (0 on the x-axis) the scores for the player vary, caused by the variance in performance across all tanks. For latencies above 0, there is a general "fan" shape depicted by the trend lines from left to right, with the smaller tank games on the bottom and the larger tank games on the top. This shape suggests that the games that have actions that require more precision (i.e. smaller tanks) are more affected by latency than are games that require less precision.

Figure 14 shows the analysis of the data to evaluate latency and deadline. As before, the x-axis is the latency, the y-axis is the score and each data point is the score achieved by the player with induced latency. The trend lines are for games with different bullet speeds. For latencies above 0, there is again a general fanning of the trend lines going from left to right, with the faster bullet speed games on the bottom and the slower bullet speed games on the top. This shape suggests that the games that have actions that require more precision (i.e. smaller tanks) are more affected by latency than are games that require less precision.

Combinations of bullet speed and tank size were run for the extremes (0.25x and 4x for each) in order to observe any compound effects. For each run, the lagged tank was given the extreme latency of 1000 milliseconds. Table 4 presents the results, setup to

[5]http://my.bzflag.org/w/Download#System_Requirements

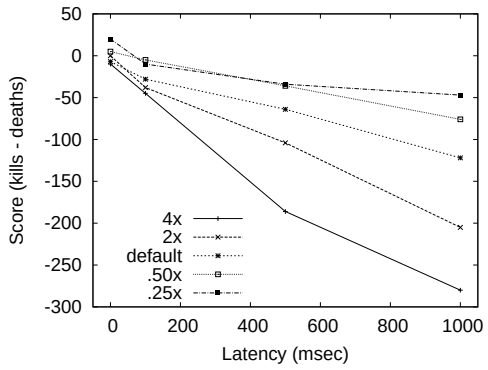

Figure 14: Score versus Latency for Different Deadline Requirements (i.e. Bullet Speeds).

align with Figure 8 in that actions that are the most sensitive to latency (high precision and tight deadlines) are in the bottom left and actions that are less sensitive to latency (low precision and loose deadlines) are in the top right. From the table, actions that have a higher precision and tighter deadline (bottom left) result in a lower score for the lagged player than actions that have a lower precision and looser deadline (top right).

		Tank Size (precision)	
		0.25x	4x
Bullet Speed (deadline)	0.25x	-71	-51
	4x	-201	-161

Table 4: Scores for Combinations of Precision (i.e. Tank Size) and Deadline (i.e. Bullet Speeds). In all cases, the lagged tank has a latency of 1000 milliseconds.

8. CONCLUSIONS

With the growth in network capacities and broadband access to the home has come the proliferation in development and deployment of multi-player, online games. Since most online games have a relatively low bitrate and are somewhat resistant to network loss, network latency is often the primary bottleneck for acceptable online game performance.

However, not all player-to-game interactions are equally sensitive to latency. In particular, the online games phases of setup, synchronization and transition are quite resistant to network latencies. Unfortunately, the play phase, where players respond to the game state and have their interactions communicated to other players, can be quite vulnerable to network latencies.

Within the play phase, different online game genres are impacted differently by network latencies. For example, first person shooter games often require quick hand-eye coordination making them sensitive to deviations in information sent to or from a game server, while real-time strategy games require relatively more thought than speed, making player actions more resilient to latency. Previous work [3] has looked at the relative impact of latency across game genres. This paper provides a general model for viewing these genres based on a classification that emphasizes the player interaction model (*avatar* or *omnipresent*) and the player perspective (*first person* or *third person*).

However, this game classification really only provides tendencies for the effects of latency on the specific game. Even within a single game genre not all actions are uniform since, for example, in a first person shooter a player may lob a grenade rather imprecisely at one moment and shoot a sniper rifle with pinpoint accuracy during the next. The exact effects of network latencies on online games depends upon two properties core to each player action: the *precision* required to complete the action successfully and the *deadline* by which the action must be completed. Actions with higher precision and tighter deadlines are more sensitive to latencies than actions with lower precision and looser deadlines.

To validate the proposed precision-deadline model, a multi-player online game, BZFlag, was modified to enable measurement of player performance with controlled amounts of latency. Modifications of tank size and bullet speed allowed for variation in the precision and deadline requirements of player actions. Hours of experimental data shows lagged players fighting with small tanks (higher precision) and fast bullets (tighter deadline) have lower scores than similarly lagged players with large tanks (lower precision) and slower bullets (looser deadline). Overall, these results provide quantifiable data that support the precision-deadline model.

The work presented here should be useful for online game designers choosing latency compensation techniques, network designers seeking to provide appropriate QoS for online games, and game players making informed choices about their Internet connections.

Our ongoing work studies how the precision-deadline model is impacted by latency compensation techniques. Latency compensation techniques are designed to ameliorate the effects of network latency on online game play by adjusting the timing and playout of game actions based on measured latency. In general, latency compensation techniques should shift game actions further from the origin (bottom left) of Figure 8, but the distance shifted may not be uniform across actions.

Acknowledgments

A special thanks to Chris Burgess and Nathan Roy for their modifications to the BZFlag code and running of the experiments.

9. REFERENCES

[1] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proceedings of ACM NetGames*, Portland, OG, USA, Sept. 2004.

[2] M. 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, Sept. 2005.

[3] M. Claypool and K. Claypool. Latency and Player Actions in Online Games. *Communications of the ACM*, 49(11), Nov. 2006.

[4] M. Dick, O. Wellnitz, and L. Wolf. Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In *Proceedings of ACM NetGames*, Hawthorne, NY, USA, Oct. 2005.

[5] T. Fritsch, H. Ritter, and J. H. Schiller. The Effect of Latency and Network Limitations on MMORPGs: a Field Study of Everquest 2. In *Proceedings of ACM NetGames*, Hawthorne, NY, USA, Oct. 2005.

[6] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE Infocom*, Hong Kong, China, Apr. 2004.

[7] A. Rollings and E. Adams. *On Game Design*. New Riders, 2003. ISBN: 1-5927-3001-9.