A Taxonomy for Player Actions with Latency in Network Games

Mark Claypool, Tianhe Wang, and Mcintyre Watts
Computer Science and Interactive Media & Game Development
Worcester Polytechnic Institute
Worcester, MA 01609, USA
{claypool|tianhe6|mlwatts}@wpi.edu

ABSTRACT

The degradation of player performance in network games with latency is well documented. However, quantifying the effects of latency on player individual actions is an unmet challenge. Under constrained bitrate conditions, player actions delayed on the client add additional latency, so network game developers need tools to help prioritize the sending of player actions. This paper presents a taxonomy for player actions with latency in network games, where player actions are defined by their precision, deadline & impact. The effects of latency along each dimension of the taxonomy is analyzed through extensive experiments with a custom 2d game. Efficacy of the taxonomy in game development is illustrated by experiments that show improved player performance when prioritizing player actions based on their expected impact derived from the taxonomy dimensions.

1. INTRODUCTION

The growth in capacity and connectivity of computer networks has increasingly enabled computer games to connect multiple players through cooperative or competitive online play. Capacities have even become sufficient for the emergence of cloud-based games where both single- and multiplayer games are played and rendered on remote servers and clients only send player commands up to the cloud.

The responsiveness of network games are affected by the network latency since, under most architectures, a player's action must be transmitted by the client to an authoritative server, acted upon and the result transmitted back to the client before the outcome of the player's action is realized. An increase in network latency means a decrease in the responsiveness of the network game and a decrease in player performance. Studies have shown network latencies can decrease player performance by 5% to 50% for every 100 milliseconds of latency for traditional network games and a 25% decrease in player performance a for every 100 milliseconds of latency for cloud-based games.

While traditional network games require only modest server bitrates [3], and cloud-based games require video-type bitrates down to the client [9], bitrates up from the client are often limited by capacity constraints. The uplink connection from a residential client is often asymmetric, smaller than the downlink. Moreover, the uplink must be shared by other in-game features, such as voice and chat commands by players, as well as in game collection and transmission of player actions and statistics data. This means that packets containing player actions cannot always be sent immediately, but instead are queued for transmission at the client based on uplink bitrate limitations. Network game systems, such as Demonware,[1] provide clients with uplink capacity estimates, as well as network latency estimates, so that games can better manage limited network resources.

The limited network resources, shared amongst all the game subsystems, means the core gameplay itself is often left with a fixed, limited budget for how many packets it can send per second. These packets are primarily player gameplay actions. Previous research has shown that not all player actions are equally sensitive to latency [7]. To explain these differences, we have proposed [6] a taxonomy of the effects of latency on different player actions based on two properties: 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 network latencies. It follows from the limited network resources and disparity in the impact of latencies on player actions, to mitigate the effects of latency on player performance game clients must prioritize their network data, deciding on a transmission order for player actions to be sent out.

Unfortunately, for game developers, there are currently no practical techniques to prioritize player actions and help determine packet order. While precision and deadline are useful for better understanding the impact of latency on actions, as proposed, the dimensions are unit-less and are not comparable with each other. Moreover, even as sensitivity to latency is explained by the taxonomy, there is no way to quantify the effects of added delay to a player action. For example, if a game client has two player actions, A and B, to transmit, with a gap of 50 milliseconds between them due to bitrate limitations, there is no easy way to quantify, and then compare, the impact on delaying A by 50 milliseconds versus B by 50 milliseconds.

In this paper, we propose to extend our precision-deadline

Technical Report WPI-CS-TR-07-14, Computer Science Department, Worcester Polytechnic Institute, December 2014. Online at: ftp://ftp.cs.wpi.edu/pub/techreports/pdf/14-07.pdf

[1] https://www.demonware.net

technology with a third dimension, called *impact*, that incorporates the effect the player action has on the game world. Then, we use the model as the basis for measuring the impact of network latency on player actions. Careful experiments with a computer game that simulates latency allows us to measure, analyze and quantify the effects of latency on player actions based on their precision, deadline and impact. Analysis of the results shows latency degrades player actions approximately linearly, with an exponential degradation fitting slightly better. Player actions with high precision are more readily impacted by latency, player actions with a loose deadline are more resistant to latency, and player actions are scaled linearly based on their impact.

The analysis also provides a blueprint for other network games to follow, providing a means to prioritize transmission of player actions for many games, thus allowing them to better mitigate the effects of latency on player actions as compared to non-prioritized transmissions. Detailed experiments with thousands of hours of simulated gameplay show the efficacy of our approach, with prioritized player actions providing for up to 100% better performance over non-prioritized player actions. This improvement increases with higher action impact and precision, but is independent of deadline.

The rest of this paper is organized as follows: Section 2 provides related work; Section 3 introduces our expanded taxonomy; Section 4 describes the methodology used to evaluate the proposed taxonomy; Sections 5 and 6 detail experiments to measure expected impact and player performance, respectively; and Section 8 summarizes our conclusions and mentions possible future work.

2. RELATED WORK

The effects of latency on traditional games has been studied for many game genres, including car racing [13], role playing games [11], and first person shooters [1]. While such work has helped better understand the impact of latency on traditional games, the results have generally not analyzed the effects of latency on specific actions but have instead treated the games as a whole.

Recent efforts have focused on latency and cloud-based games, measuring the responsiveness of a cloud-based gaming platforms with added latency [15, 4], and conducting user studies measuring the effects of latency on cloud-game players [12, 8]. Similar to earlier studies with traditional games, the results have generally not analyzed the effects of latency on specific actions.

Lower level studies have examined the effects of latency on user interactivity for a variety of game-like tasks, such as target acquisition [14], or have showed gamers have more refined temporal processing [10]. While helpful to better understand human sensitivities to latencies for interactive tasks, the results were not been applied to computer games.

Our previous work has isolated player actions and explored the effects of latency for real-time strategy games [5] and first person shooter games [2], and has attempted to classify player actions with regard to latency sensitivity [6]. However, such work has not generalized the effects of latency on player actions, nor yet led to a method to apply the results in game development.

3. PLAYER ACTION TAXONOMY

For traditional network games, when a player performs an action, the client sends a message to the server, which processes the action and sends any changes to the game state back to the client, which renders the game on the local display. Cloud-based games do the same, except that the server renders the game and sends the frames down as video.

All player actions in this client-server architecture are delayed by the round-trip latency between the client and server. If the delay 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 can also degrade player performance. How much the player's action (and hence performance) is impacted by the delay is determined by the deadline and precision requirements of a given player action.

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*, deadline for a portal spell is the time it takes for an avatar to read a magic scroll and invoke a portal that will transport the avatar back to town. Precision is the degree of accuracy required to complete the interaction successfully. For example, in *Call of Duty* precision is the accuracy required to shoot a distant enemy with a sniper rifle. Different player actions have disparate deadline and precision requirements. For example, moving an avatar in a computer game requires high precision but typically 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.

As first presented in [6], Figure 1 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 precision. The shooting sniper in the bottom left has high precision and a tight deadline, building in a construction game in the bottom right has a high precision but a loose deadline, and third-person combat in the middle has a lower precision than either sniper shooting or building, and a looser deadline than sniping but a tighter deadline than building. 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, sniping and racing are sensitive to latency, while casting an area spell and explore are less sensitive to latency.

While the taxonomy is descriptive for most player actions and can help categorize genres of games (as indicated by the "Avatar First Person", "Avatar Third Person" and "Omnipresent" labels in Figure 1), by itself it is not sufficient for prioritizing player actions in order to mitigate the effects of latency while a game is running. In particular, the axes are of different units, so are difficult to compare quantitatively, even if they could be normalized. Moreover, the taxonomy, as stated, does not account for the effect the player action has on the game world. This means a priority transmission based solely on the distance of an action from the origin may transmit a "more important" action after a "less important" action.

For example, consider an arcade-style game where a player has two weapons – Weapon A fires fast, precise bullets and Weapon B fires slow bombs that explode in a wide area. The fast, precise bullets place Weapon A actions close to the ori-

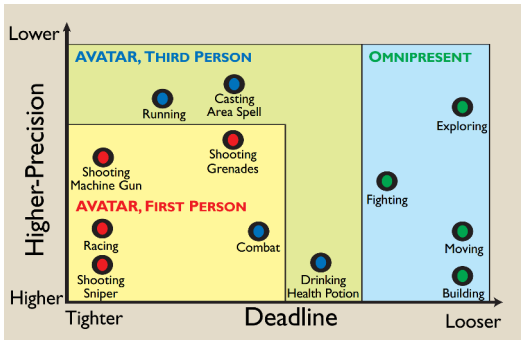

Figure 1: Taxonomy of Different Player Actions along the Precision and Deadline Axes.

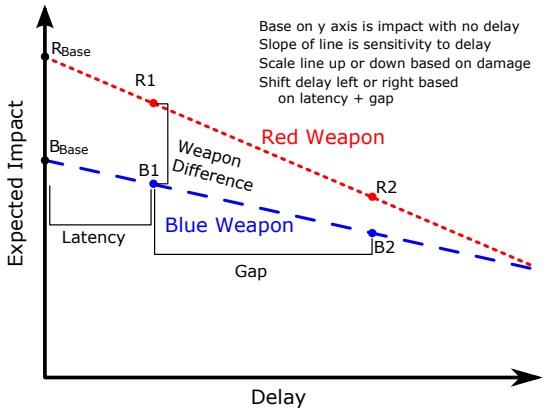

Figure 2: Expected Impact

gin in Figure 1, and the slow, low precision (given the area of effect) bullets place Weapon B actions farther from the origin. Weapon A actions, being thus more sensitive to latency would seem to be prioritized first. However, suppose the damage from the Weapon A bullets was far less than the damage from Weapon B bullets. In this case, prioritizing a Weapon A action over a Weapon B action may mean the Weapon B bullet misses the target, thus impacting the player performance more.

Thus, we propose to extend the taxonomy of player actions by a third dimension called *impact*. Impact is the effect that the player action has on the game world. For example, in the case illustrated above, impact is the amount of damage a bullet fired from a weapon causes when it hits a target. Used all together, precision, deadline and impact allow for a representative prioritization of player actions that when followed for network transmissions, sends the more important player actions first, thus mitigating the effects of latency on player performance.

To illustrate the use of the Taxonomy, consider an arcade-style game where the player has various weapons and shoots projectiles (bullets) at opponents. As is typical of many such games, weapon bullets have different speeds, areas of effects and damage dealt.

The hypothesized effects of delay on player actions in such a game is depicted in Figure 2. The x-axis is the delay for player actions, which is the round-trip time from the client to the server plus any added time due to queuing on the client during bandwidth limitations. The y-axis is the expected impact of the action, which, for this game, is how much damage the weapon is expected to do. The expected impact depends upon how often the bullet hits an opponent.

The delay value on the x-axis is the sum of the network latency and any added gap due due bitrate limitations.

A weapon (and the bullets it fires) is represented by a downward trend, show as a line in Figure 2. Each weapon has a different line, with a different slope and a different y-intercept depending upon the bullet speed, area of effect and damage. The graph depicts Weapon 1 (red, small dashes) and Weapon 2 (blue, large dashes) as two different lines.

To determine the impact of a weapon, the delay provides the x value and the line equation for the weapon (e.g., $y = ax + b$) provides the y, or expected impact value.

For Weapon 2, the horizontal distance between $B1$ and B_{base} indicates the amount of added delay from the network latency. The horizontal distance between points $B1$ and $B2$ indicates the amount of added delay the game system may have added by adding a required gap between packets.

The slope of the line indicates the sensitivity of the weapon to delay. The steeper the slope, the more sensitive it is to delay.

The y-intercept indicates the "base" expected impact of the weapon – the expected impact if there were no delay and the weapon's bullet was fired immediately. The higher the base, the greater the expected impact.

The base, and all points on corresponding line, are scaled higher or lower depending upon the weapon's damage. For example, increasing the damage by 5 multiplies all points on the line by 5 (e.g., $y = 5 \times (ax + b)$).

4. METHODOLOGY

This section describes the methodology used to evaluate the efficacy of the proposed taxonomy:

1. Build a computer game that allows for configuration of player actions and control of delay (Section 4.1).
2. Run experiments to measure the expected impact of player actions based on precision, deadline & impact (Section 5).
3. Analyze the results and derive a quantified, algebraic model for predicting the expected impact versus latency for each player action.
4. Run experiments use the model in priority queue, evaluating the benefit to player performance over an unordered queue (Section 6).

4.1 Saucer Hunt

In order to test the efficacy of the proposed Taxonomy in a controlled fashion within a computer game, we designed and developed a game that provided the necessary game for evaluation. For ease of development and customization, we used the Dragonfly[2] game engine (version 3.4). The Dragonfly project is designed to teach aspiring students about game engine development, with students building a game engine from scratch, and, in turn, use their own engines to make games. The Dragonfly engine is a fully functional

[2]`http://dragonfly.wpi.edu`

game engine, with game loop control, game world management, game object physics, user input management, and 2d graphics support, including animated sprites.

We call our game *Saucer Hunt*, based on the popular Nintendo game *Duck Hunt*. Figure 3 depicts a screenshot. The player commands a space ship at the bottom of the screen and shoots at enemy saucers that fly horizontally across the top of the screen. The bullets explode when reaching the top of the screen, destroying saucers that are enveloped in their explosion. Points are obtained based on how much damage the bullet does to the saucer.

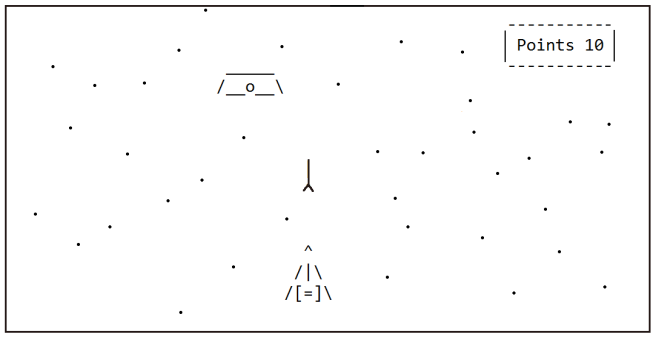

Figure 3: Saucer Hunt screenshot

There is only one saucer at a time on the screen. The saucer starts at either the left or right side of the screen and attempts to fly to the opposite side, moving at varying speeds to make it more difficult to shoot. The player's ship can only fire one bullet per weapon per saucer - i.e., if a bullet misses, the saucer must reach the edge of the screen before the weapon can fire again. A game last for 90 seconds, with the player trying to score as many points as possible in that time.

As designed, Saucer Hunt is customizable so as to allow exploration of the space of precision, deadline & impact. The bullet explosion provides an area of effect that is the precision. The bullet speed determines how fast the bullet reaches the target and explodes, so is the deadline. The amount of damage a bullet's explosion that hits the target causes is the impact. The number of weapons can be customized, with each weapon's bullets having a different area of effect, speed, and damage.

Saucer Hunt can be configured to simulate both network latency and bitrate limitations. Network latency is simulated by delaying player commands to fire by a fixed amount of time. Bitrate limitations are simulated by imposing a minimum gap time between player commands. Both kinds of delay, network latency and gap, are in units of the game loop – e.g., with a game loop time of 33 milliseconds, a network latency of 3 adds 124 milliseconds of latency to the player commands.

In order to Saucer Hunt easier to run in an automated fashion, an AI player was created to emulate a human player. The AI player determines when to fire a bullet based on observation of the saucer's speed and distance from the target. Pilot studies suggest the AI does comparable, but a bit better, than most human players.

The Dragonfly engine allows games to be run in "headless" mode, as well as provides control over the game loop, letting Saucer Hunt run as fast as possible. This compresses a 90 second game session with AI to less than one second and allows repeated game sessions to be run in the background. When a game is complete, Saucer Hunt reports player statistics, including the accuracy and points for each weapon, as well as the total points the player scored.

The Saucer Hunt source code is available for download via its git repository, with a link at: `http://www.cs.wpi.edu/~claypool/papers/expected-impact/` Building Saucer Hunt requires Dragonfly, which can be downloaded from the Dragonfly engine Web page: `http://dragonfly.wpi.edu/engine/`

5. EXPECTED IMPACT

Experiments with Saucer Hunt were run for a single weapon over an exhaustive set of player actions. The player was AI controlled, the game set to "headless" and the frame time set to 0 in order to run the experiments in the background. All combinations of the weapon for speeds 0.25, 0.5, 0.75, 1, 2, 3 and 4, and areas of effect 0, to 10 were tested, a total of 77 combinations. For each combination, delays from 0 to 990 milliseconds were tested in steps of 33 milliseconds (one game loop). For each weapon configuration at each latency, one-thousand games were played for each combination. In all, about sixty-thousand hours of gameplay were emulated, or nearly seven years straight of playing Saucer Hunt.

5.1 Results

When a weapon with damage 1 is fired, the expected impact is the chance that the weapon will hit at that latency – i.e., the weapon's accuracy. This value is scaled up for damages greater than 1. All weapon configurations were analyzed with respect to latency over the range tested, doing both linear regression and exponential fits to the data. As expected, the expected impact for all weapons have a downward trend i.e., the expected impact decreases with an increase in latency. 80% of the weapons have a linear correlation of -0.9 or stronger and 95% are -0.8 or stronger. The weakest correlations, around -0.6, are all slow bullets (speeds less than 1) that require high precision (AoE 0). Exponential fits showed only slightly better fits for only high speed weapons (see Section 7).

Due to space constraints, Sections 5.2-5.4 show a subset of the analysis is shown, choosing results that best typify weapon precision (area of effect), deadline (speed) & impact (damage).

5.2 Damage

Figure 4 depicts the expected impact versus delay for two weapons that have the same speed (0.5) and area of effect (7) but differ in their damages, 1 and 3. The x-axis is delay in milliseconds and the y-axis is the expected impact in damage. Each point is the average of 1000 experimental runs with the lines showing a linear regression fit through the data points. Both weapons show a downward trend in expected impact with an increase in delay. The expected impact of the damage 3 weapon is higher than that of the damage 1 weapon since when a higher damage weapon hits, it does more damage. In fact, the line for the damage 3 weapon is scaled 3 times (i.e., the y values are multiplied by 3) over that of the damage 1 weapon. This both increases the y-value (expected impact) and also makes the slope of

the higher damage weapon steeper. The higher the weapon damage, the greater the sensitivity to delay.

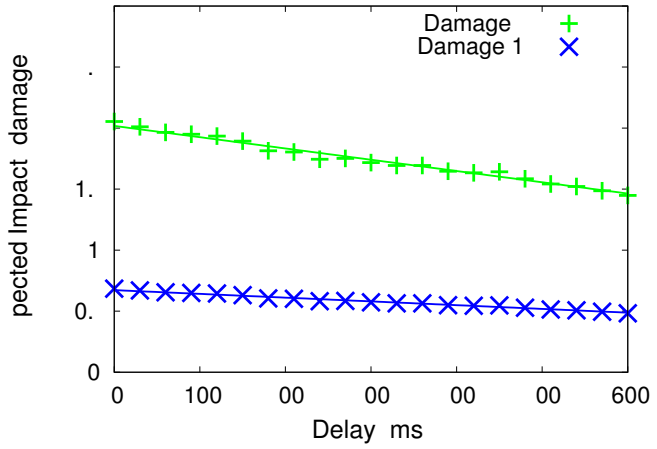

Figure 4: Expected Impact versus Damage

5.3 Speed

Figure 5 depicts the expected impact versus delay for two weapons that have the same area of effect (7) and damage (1) but differ in their speeds, 0.25 and 4. The axes, data points and trendlines are as for Figure 4. Both weapons show a downward trend in expected impact with an increase in delay. The expected impact of the speed 2 weapon is higher than that of the speed 0.25 weapon since it is easier for the player to hit the target with a higher speed weapon. The higher speed weapon also has a steeper slope, meaning it is more sensitive to delay than the speed 0.25 weapon.

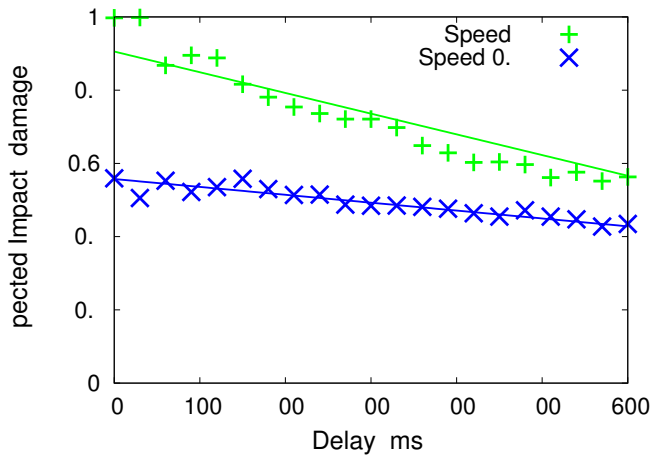

Figure 5: Expected Impact versus Speed

5.4 Area of Effect

Figure 6 depicts the expected impact versus delay for two weapons that have the same speed (0.5) and damage (1) but differ in their areas of effect (AoE), 3 and 7. The axes, data points and trendlines are as for Figure 4. Both weapons show a downward trend in expected impact with an increase in delay. The expected impact of the AoE 7 weapon is higher than that of the AoE 3 weapon since a larger area of effect is likely to do more damage. Moreover, the slopes of the lines are nearly the same. Thus, any added delay to fire actions degrades the impact of each weapon by the same amount.

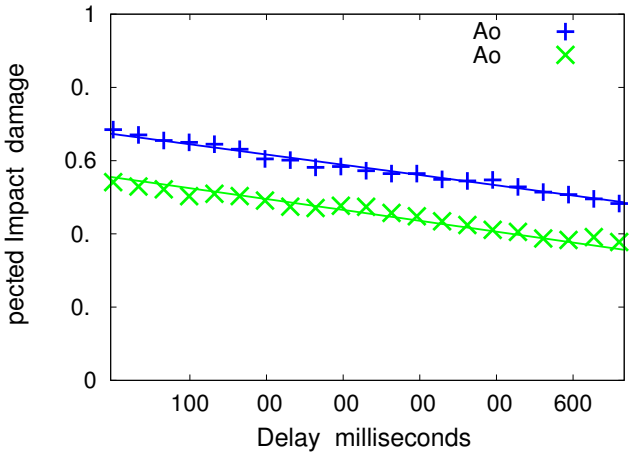

Figure 6: Expected Impact versus Area of Effect

6. MITIGATING LATENCY

This section describes how the precision, deadline & impact taxonomy presented in Section 3 can be used to mitigate the effects of latency on player performance.

6.1 Predicting Expected Impact

For a game with multiple weapons, when a weapon is fired and there is a fire command for another weapon already in the queue, the game has a choice for which player command to transmit first. In order to minimize the impact of delay on player performance, the command ordering should try to maximize the total expected impact.

For example, consider a game with a small amount of network latency and a large bitrate limitation (i.e., a big gap between packets) and two weapons. These weapons are depicted by the two downward sloping lines in Figure 2. Assume the player fired the Red Weapon (W_R) and the player action is in queue ready to transmit. The queue is:

1. W_R (delay = latency)

In Figure 2, point $B1$ represents the expected impact computation using the W_R line for this weapon. At this time, the player fires the Blue Weapon (W_B) with the command for W_R still in queue. The game has two choices:

A) W_B can be placed after W_R in the queue. Since packets need to be separated by at least a gap delay, the queue would look like:

1. W_R (delay = latency)
2. W_B (delay = latency + gap)

The impact of W_B at this point is shown with point $R2$ in Figure 2.

Or:

B) W_B can be placed before W_R, adding a gap to the delay for W_R. This queue would look like:

1. W_B (delay = latency)
2. W_R (delay = latency + gap)

The impact of W_B at this point is shown with point $R1$ in Figure 2 and the impact of W_R at this point is shown with point $B2$.

Since the goal is to maximize the total expected impact, the better choice A) or B), depends upon which is larger:

- Impact of W_B at $R2$ + Impact of W_R at $B1$
- Impact of W_B at $R1$ + Impact of W_R at $B2$

The larger of the two should be chosen for the corresponding queue order. Based on Figure 2, the second option is considerably smaller than the first, thus W_B should be placed before W_R. Note that a first-in, first-out system or one that was otherwise ignorant about the impact of latency on player actions would have chosen sending out W_R before W_B, thus resulting in a greater degraded player performance.

In summary, handling outgoing player commands in a first-in, first-out (FIFO) manner can result in a greater degradation on player performance than a priority queue. In order to decide upon the best order for player actions, the relationship between precision (area of effect), deadline (speed) & impact (damage), and delay must be known – i.e., there needs to be an equation, such as a line, for expected impact versus delay for each player action in the game. Then, when a player takes an action, the game should compute the expected impact for placing the command at each spot in the queue, re-ordering the queue to maximize the total expected impact.

6.2 Experiments

We ran experiments to measure the efficacy of using knowledge of the precision, deadline & impact of player actions in Saucer Hunt.

For evaluation, Saucer Hunt was configured to have two weapons. The first weapon always fires bullets with damage 1, speed 0.25, and area of effect 4. The second weapon has the same base characteristics as the first weapon, but varies one of the characteristics for each experimental run: damage 1, 2, 4, 8, and 16; speed 0.25, 0.75, 1.25, 2 and 4; and area of effect 4, 6, 8, 10 and 12.

The typical uplink for an role-playing game or real-time strategy game is about 10 packets/second [16, 5]. Assuming this rate is imposed by a capacity limitation, this suggests a minimum gap between subsequent packets of about 100 milliseconds. Thus, a minimum gap of 99 milliseconds (the equivalent 3 game loops) between packets is used for all experiments.

Internet latencies can vary, but 2014 data on over 28 million game sessions from the popular game League of Legends found that the most common ping time for players is around 50 milliseconds [17]. Thus, a network latency of 66 milliseconds (the equivalent 2 game loops) is used for all experiments.

Saucer Hunt was extended with implementations for two different types of outgoing queues for player actions: a) a FIFO queue that sends player actions without regard to expected impact, and b) a priority queue that orders outgoing player actions so as to maximize total expected impact. For the priority queue, Saucer Hunt used the linear prediction for each weapon in computing the expected impact.

For each queue type and each weapon configuration combination, Saucer Hunt was run 100 times, computing mean values with 95% confidence intervals.

Figure 7 depicts the result. The y-axis for all three graphs is the player's performance, the total points, over a 90 second game. The points in each graph is the average total points achieved by the player across all games at that configuration, shown with 95% confidence intervals.[3] Each graph has two trendlines – the blue '*' uses a first-in, first-out (FIFO) queue and the green 'x' uses a priority queue where player commands that are separated by a gap are enqueued so as to maximize the total expected impact. The top graph varies the area of effect (AoE) of bullets fired from the second weapon, from the same AoE as the first weapon (4) to 3 times the AoE. The middle graph varies the speed of bullets fired from the second weapon, from the same speed as the first weapon (0.25) to over 30 times faster than the first weapon. The bottom graph varies the damage of bullets fired by the second weapon, from the same damage as the first weapon (1) to 16 times the damage as the first weapon.

From Figure 7 top, prioritizing player actions based on differences in weapon AoE does not have much benefit to player performance. As seen from Figure 6, while AoE does change expected impact, weapons with different AoEs have similar slopes. Thus, the total points a player receives is not changed based on prioritizing bullets from one weapon with a different AoE over another.

From Figure 7 middle, prioritizing player actions based on differences in weapon speed does not have much benefit to player performance. This is somewhat surprising given the results in Figure 5 that shows weapon with different speeds have different slopes, therefore considering their expected impact when transmitting should result in higher player performance. However, the lack of a difference can be explained by the Saucer Hunt game itself. Since the player fires weapons with slower speeds sooner than weapons with faster speeds, effectively leading the Saucer more the slower the weapon is, by the time the second weapon is fired, the outgoing queue has cleared. In other words, there is no queue to no opportunity for optimization. For games with a higher rate of player actions, especially those where the player takes several actions simultaneously, prioritizing player commands based on speed should show some benefit.

From Figure 7 bottom, prioritizing player actions based on differences in weapon damage can have a big impact on player performance. This is explainable from Figure 4 that shows the expected impact of a weapon is scaled directly with damage. Prioritizing weapons that do more damage over those that do less damage makes it more likely the high damage bullets hit, thus improving player performance.

Overall, it is most important to prioritize player actions based on the slope of their expected impact versus delay. When player actions have the same slopes, differentiating between them is not important – basically, player perfor-

[3]The confidence intervals are so small for most points as to be indiscernible.

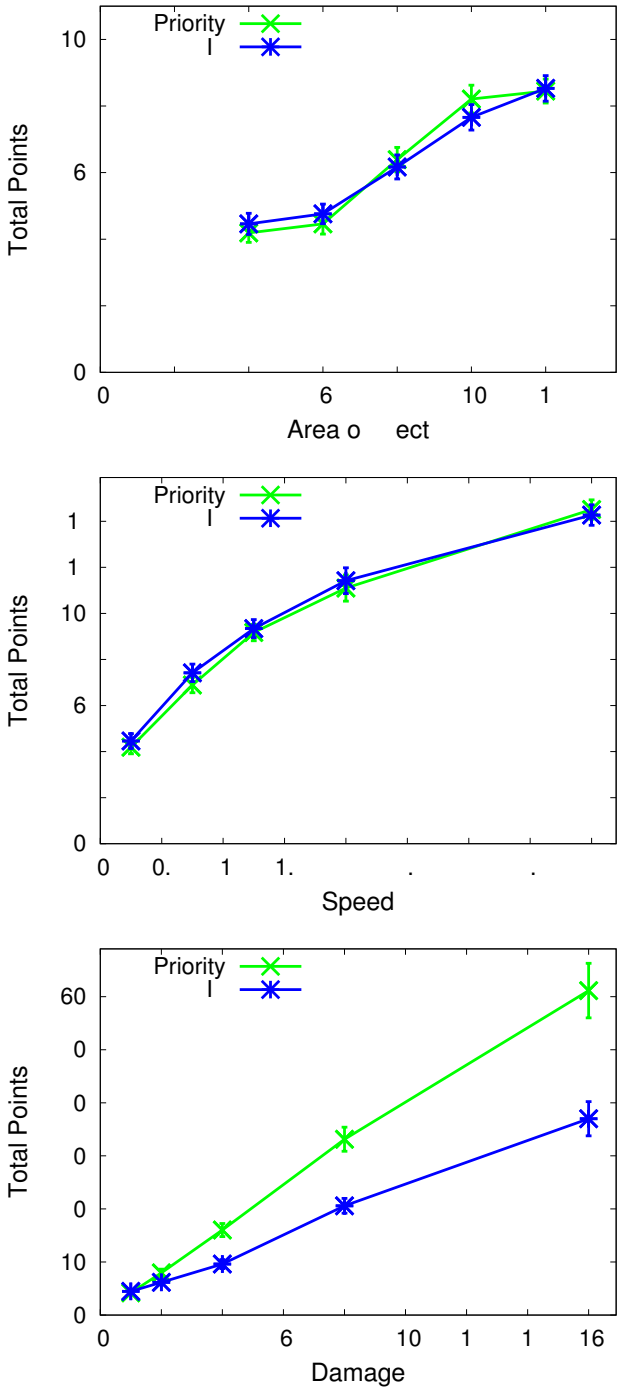

Figure 7: Player Performance

mance is the same no matter which action is prioritized. When player actions have different slopes, actions that have a steeper slope are more sensitive to latency and should be favored.

7. DISCUSSION

While linear fit works for most player actions in Saucer Hunt, as evidenced by the relatively high correlation coefficients, there are regions for many of the player actions that may not be well-represented by a diagonal line. For example, consider the graph of expected impact versus delay for a weapon with high speed (4) bullets with moderate precision (AoE 4) in Figure 8. For latencies under 30 milliseconds, the expected impact is 1 (the bullets always hit). From 30 to about 150 milliseconds, the expected impact decreases in exponentially. From 150 milliseconds and beyond, the expected impact decreases approximately linearly over the rest of the measured range. While the correlation coefficient for a linear fit is still good (-0.9), fitting a line to the entire range does not match the latency well for some regions. Prioritizing player actions for these areas may result in degraded player performance versus more accurate models.

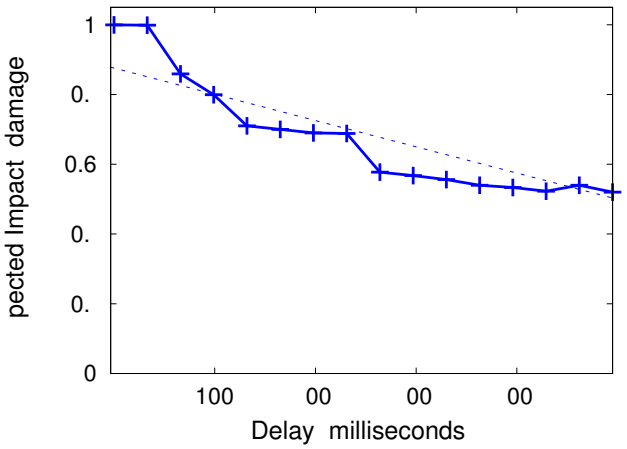

Figure 8: Expected impact with linear fit

Ideally, the results here quantifying the expected impact of delay on precision (area of effect), deadline (speed) and impact (damage) could be generalized for other games beyond Saucer Hunt. Doing so is left as future work. While exhaustively measuring the impact of delay on all player actions is not feasible for most games, the results from Saucer Hunt point to a possible method useful for most games. Basically for each player action: 1) Determine the expected impact with no delay – this is the "base" or y-intercept for the player action, depicted in Figure 2; 2) Measure the expected impact of the action with a fixed amount of delay – this is one point along the line, such as point B2 for the Blue Weapon in in Figure 2; 3) construct a linear from the line connecting the base and measured point that can be used to compute the expected impact of that action in the presence of delay – results over an exhaustive set of player actions and delays show that a linear relationship well-represents the impact of delay on player actions in most cases.

The relationship between player actions and latency is not

linear in all cases. In fact, as suggested by [6] and data on Saucer Hunt with high speed weapons, actions that are the most sensitive to delay degrade exponentially with latency. In such cases, an exponential curve likely fits it better. However, for many actions over many latency ranges, a linear fit well-represents the degradation to performance for player actions. Since a linear fit is easy to construct (two points make a line) and understand, and models should be parsimonious (explain the relationship with as few predictor variables as possible), using a line to represent the impact of latency on player actions has merits.

For Saucer Hunt, for most actions tested over a range of parameters and delays, a line works quite well with an exponential fit only slightly better. It is reasonable to assume these results will hold for other games, particularly those most similar to Saucer Hunt – i.e., arcade-style games with projectiles. Where the relationships may differ significantly are for those games with different kinds of actions (e.g., moving an avatar or melee combat).

8. CONCLUSION

The study of the effects of latency on network and online games is increasingly important with the growth in network games and cloud-based gaming. While the fact that latency degrades player performance for network games is well-understood, the exact relationship between latency and player actions is neither well-understood nor appropriately quantified. Specifically, quantitative relationships between player actions and latency are needed in order for game systems with bitrate constraints to prioritize player actions so as to mitigate the effects of latency.

This paper takes a step towards better understanding and quantifying the effects of latency on player actions and latency. The contributions include: 1) a taxonomy of player actions and latency, based on precision, deadline & impact; 2) illustration of the taxonomy through experimentation with a computer game, quantifying the relationship between area of effect (precision), speed (deadline) and damage (impact) and latency; 3) use of the experimental results in a priority queue system, illustrating how the effects of latency on player actions can be mitigated for game systems with network bitrate constraints.

For the game tested, a 2d arcade style shooting game, the expected impact degrades linearly with latency. The steeper the slope, the more sensitive the player action is to latency. The slope is affected most dramatically by the impact, represented as the damage a weapon inflicts on the opponent. The slope is also affected by the deadline, represented as the speed of the projectiles, with tighter deadline actions being more sensitive than looser deadline actions. The slope is not affected by the precision, represented as the area of effect of the projectiles, but the base (y-intercept) is, with lower-precision weapons having a higher base. For game systems with network latency and bitrate constraints, prioritizing player actions using the taxonomy of precision, deadline & impact showed about a 150% improvement to player performance.

The approach used in this paper was applied to a specific game, but the results may generalize to other games. Application of the methodology used in this paper to another game left as future work. In addition, developing a general model for the effects of delay on player actions without necessarily specifically measuring a working game is also possible future work.

9. REFERENCES

[1] G. Armitage. An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3. In *Proceedings of the 11th IEEE International Conference on Networks (ICON)*, Sydney, Australia, Sept. 2003.

[2] 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 Network and System Support for Games Workshop (NetGames)*, Portland, OG, USA, Sept. 2004.

[3] W. chang Feng, F. Chang, W. chi Feng, and J. Walpole. Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, Nov. 2002.

[4] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu. On the Quality of Service of Cloud Gaming Systems. *IEEE Transactions on Multimedia*, 26(2), Feb. 2014.

[5] 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.

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

[7] M. Claypool and K. Claypool. Latency Can Kill: Precision and Deadline in Online Games. In *Proceedings of the First ACM Multimedia Systems Conference (MMSys)*, Scottsdale, Arizona, USA, Feb. 2010. (Invited paper).

[8] M. Claypool and D. Finkel. The Effects of Latency on Player Performance in Cloud-based Games. In *Proceedings of the 13th ACM Network and System Support for Games (NetGames)*, Nagoya, Japan, Dec. 2014.

[9] M. Claypool, D. Finkel, A. Grant, and M. Solano. On the Performance of OnLive Thin Client Games. *Springer Multimedia Systems Journal (MMSJ) - Special Issue on Network Systems Support for Games*, pages 1–14, feb 2014. DOI 10.1007/s00530-014-0362-4.

[10] S. Donohue, M. Woldorff, and S. Mitroff. Video Game Players Show More Precise Multisensory Temporal Processing Abilities. *Springer Verlag Attention, Perception & Psychophysics*, 72(4):1120–1129, 2010.

[11] 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 the 4th ACM Network and System Support for Games (NetGames)*, Hawthorne, NY, USA, Oct. 2005.

[12] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 330–335, Seoul, Korea, 2011.

[13] L. Pantel and L. 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)*, Miami, FL, USA, May 2002.

[14] A. Pavlovych and C. Gutwin. Assessing Target Acquisition and Tracking Performance for Complex Moving Targets in the Presence of Latency and Jitter. In *Proceedings of Graphics Interface*, pages 109–116, Toronto, Ontario, Canada, 2012.

[15] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud Gaming: Architecture and Performance. *IEEE Network*, 27(4):16–21, 2013.

[16] M. Suznjevic, O. Dobrijevic, and M. Matijasevic. MMORPG Player Actions: Network Performance, Session Patterns and Latency Requirements Analysis. *Springer Multimedia Tools and Applications*, 45(1–3):191–214, Oct. 2009.

[17] M. Vron, O. Marin, and S. Monne. Matchmaking in Multi-player On-line Games: Studying User Traces to Improve the User Experience. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Singapore, Mar. 2014.