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|twang6|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 are 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.

General Terms

Performance, Human Factors

Categories and Subject Descriptors

D.0 [**Software**]: Gen.; K.8 [**Personal Computing**]: Games

Keywords

Games, Latency, Taxonomy, Precision, Deadline, Impact

1. INTRODUCTION

Network games are affected by network latency since with most game 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 game and a decrease in player performance. Studies have shown network latencies can decrease player performance by up to 25% for every 100 milliseconds of latency for traditional games [4] and cloud-based games [5].

NOSSDAV'15 March 18-20, 2015, Portland, OR, USA

http://dx.doi.org/10.1145/2736084.2736093

Bitrates up from the network game 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 and game data for analytics. This means that packets containing player actions cannot always be sent immediately, but instead are queued for transmission at the client because of uplink bitrate limitations. Network game systems, such as Demonware,[1] provide clients with capacity and latency estimates, presenting the opportunity for games to better manage limited network resources.

The limited network resources 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. In an attempt to explain previous research that shows that not all player actions are equally sensitive to latency, we proposed a taxonomy of the effects of latency on player actions based on two action properties [4]: the *precision* required to complete the action and the *deadline* by which the action must be completed. This taxonomy provided a better understanding of what actions are most sensitive to latency and should be prioritized in order to mitigate the effects of latency on player performance.

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 player actions, as proposed the dimensions are not quantified nor comparable with each other. For example, if a game client has two player actions, A and B, to transmit with a required gap of 50 milliseconds between them due to bitrate limitations, the taxonomy provides no easy way to quantify, and then compare, the impact on delaying A by 50 milliseconds versus delaying B by 50 milliseconds.

In this paper, we extend our precision-deadline taxonomy with a third dimension, called *impact*, that incorporates the effect the player action has on the game world. Extensive experiments with a custom computer game allows us to measure, analyze and quantify the effects of latency on player actions based on precision, deadline and impact. Analysis of the results shows latency degrades player actions approximately linearly, with an exponential fit only slightly better. High precision actions are readily affected by latency, looser deadline actions do not change their sensitivity to latency, and actions scale linearly with their impact.

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

The analysis also provides a blueprint that other network games can follow for a means to prioritize transmission of player actions and mitigate the effects of latency. Detailed experiments with thousands of hours of simulated gameplay show the efficacy of our approach, with prioritized player actions providing better performance over non-prioritized player actions. This improvement increases with greater impact and with higher 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 game developed to evaluate the proposed taxonomy; Sections 5 and 6 detail experiments to measure expected impact and player performance, respectively; Section 7 discusses the findings; and Section 8 summarizes our conclusions and possible future work.

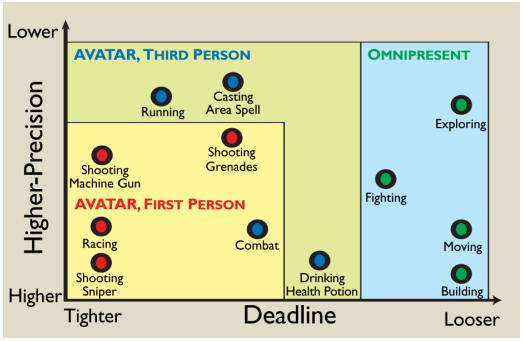

Figure 1: Taxonomy of Player Actions along the Precision and Deadline Axes

2. RELATED WORK

The effects of latency on traditional games has been studied for many game genres, including car racing [9], role playing games [7], 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 [2, 11], and conducting user studies measuring the effects of latency on cloud-game players [5, 8]. Similar to earlier studies with traditional games, the results have generally not analyzed the effects of latency on specific player actions.

Lower level studies have examined the effects of latency on user interactivity for a variety of game-like tasks, such as target acquisition [10], or have showed gamers to have more refined temporal processing [6]. While helpful to better understand human sensitivities to latencies for interactive tasks, the results have not been applied to computer games.

Our previous work has isolated player actions within games, exploring the effects of latency for specific actions (e.g., combat in real-time strategy games [3]), and has attempted to classify player actions with regard to latency sensitivity [4]. However, such work has not generalized the effects of latency on player actions, nor yet led to a method to apply the results to game development.

3. PLAYER ACTION TAXONOMY

All player actions in a client-server architecture are delayed by the round-trip latency between the client and server. How much the player's action is impacted by the latency is determined by the requirements of a given action along two primary axes, *deadline* and *precision*. Deadline is the length of time it takes to achieve the final outcome of the action, and precision is the accuracy required to complete the action successfully.

As first presented in [4], 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. In general, the further an action is from the origin in the precision-deadline plane, the lower the impact that latency has on player performance. Thus, 'Racing' is sensitive to latency, while 'Exploring' is 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 the taxonomy is not sufficient for prioritizing player actions in order to mitigate the effects of delays during a game. In particular, the axes have 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 origin in Figure 1, and the slow, low precision (given the area of effect) bombs place Weapon B actions farther from the origin. This suggests Weapon A actions, being more sensitive to latency, should be prioritized first. However, suppose the damage from a Weapon A bullet is far less than the damage from a Weapon B bomb. In this case, prioritizing a Weapon A action over a Weapon B action may mean the Weapon B bullet misses the target completely, doing no damage, thus impacting the player performance more than a Weapon A bullet that missed.

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. All together, precision, deadline & impact describe the effects of latency on player actions that when used for network transmission prioritization, sends the more important player actions first, thus mitigating the effects of delay 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 effect 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

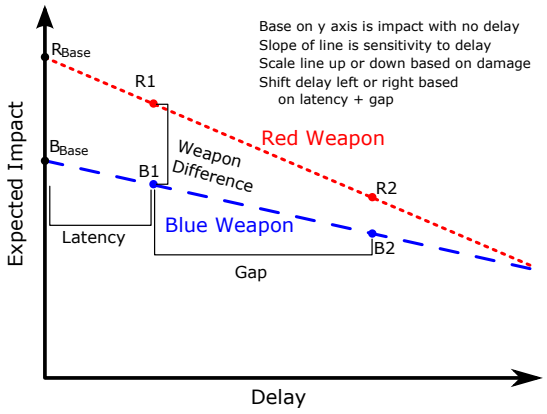

Figure 2: Expected Impact Due to Delay

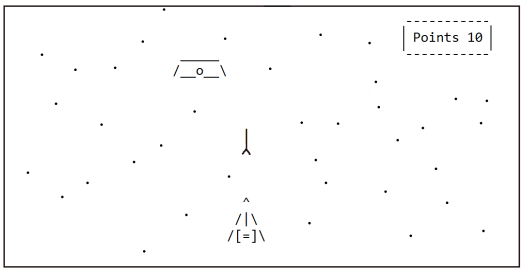

Figure 3: Saucer Hunt Screenshot

for player actions, which is the round-trip latency from the client to the server plus any added time from queuing on the client due to bitrate 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 multiplied by the damage dealt.

A weapon (and the bullets it fires) is represented by a downward trend, shown 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. Weapon 1 (red, small dashes) and Weapon 2 (blue, large dashes) are 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 the y-axis indicates the amount of added delay from the network latency. The horizontal distance between points $B1$ and $B2$ indicates the amount of added delay from the game system's bitrate constraint requiring a gap between packets.

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

The y intercept indicates the "base" expected impact of the weapon (e.g., R_{Base}) – 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 when there is no delay.

The base, and all points on the 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., $y = 5 \times (ax+b)$), thus increasing the slope.

4. SAUCER HUNT

In order to test the efficacy of the proposed taxonomy in a controlled fashion, we designed and developed a configurable computer game called *Saucer Hunt*,[2] using the Dragonfly[3] (v3.4) game engine. Figure 3 depicts a screenshot of Saucer Hunt. The player controls a space ship at the bottom of the screen and shoots bullets at enemy saucers that fly horizontally across the top of the screen. The bullets explode when reaching the top of the screen, destroying any saucer that overlaps the explosion. Points are obtained based on how much damage the bullet does to the saucer.

There is only one saucer on the screen at a time. 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 lasts for 90 seconds, with the player trying to score as many points as possible in that time.

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

Network latency is simulated by delaying player actions to fire by a fixed amount of time. Bitrate limitations are simulated by imposing a minimum gap time between player actions. Both kinds of delay, network latency and gap, are configurable in increments of the game loop – e.g., with a game loop time of 33 milliseconds, a network latency of "4" adds 132 milliseconds of latency to the player commands.

To enable running many Saucer Hunt experiments, 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, the weapon's bullet 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, allowing Saucer Hunt to 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.

Links to Saucer Hunt and Dragonfly are available at: `http://www.cs.wpi.edu/~claypool/papers/expected-impact/`

5. EXPECTED IMPACT

Experiments with Saucer Hunt were run for a single weapon over an exhaustive set of player actions. The player was AI

[2]Based on the popular Nintendo game *Duck Hunt*.
[3]`http://dragonfly.wpi.edu`

controlled, the game run in "headless" mode and the game loop time set to 0 to run in the background. All combinations of the weapon with damage 1, speeds 0.25, 0.5, 0.75, 1, 2, 3 and 4, and areas of effect 0 to 10 (a total of 77 combinations) were tested. For each combination, delays ranged from 0 to 990 milliseconds in steps of 33 milliseconds (one standard game loop). This reflects both network latency and delay within the game due to limits on simultaneous actions. For each weapon configuration at each delay, 1000 games were played. In all, over 55,000 hours of gameplay were emulated, equivalent to more than 6 years of continuously playing Saucer Hunt.

5.1 Results

When a weapon with damage 1 is fired, the expected impact is the chance that the weapon hits – i.e., the weapon's accuracy. Expected impact is scaled up (i.e., multiplied by damage) for damages greater than 1. The expected impact versus delay for all weapon configurations were analyzed over the range of delays, doing both linear regression and exponential fits of the data. As expected, the expected impacts for all weapons have a downward trend – i.e., the expected impact decreases with an increase in delay. 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 for loose deadlines (speeds less than 1) with high precision (area of effect 0). Exponential curves have only slightly better fits and only for high speed weapons (see Section 7).

Due to space constraints, Sections 5.2-5.4 show only a small subset of the analysis – 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 (accuracy × damage). Each point is the average of 1000 experimental runs with the lines showing a linear regression fit through the 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 makes the slope steeper. The higher the weapon damage, the greater the sensitivity to delay.

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 2. 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 lower speed weapon.

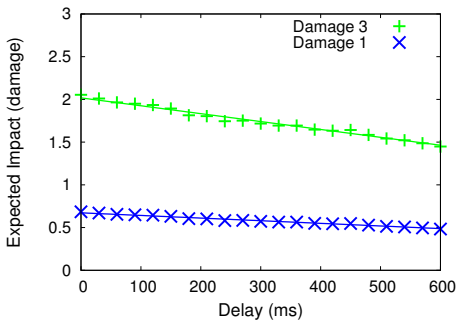

Figure 4: Expected Impact versus Damage

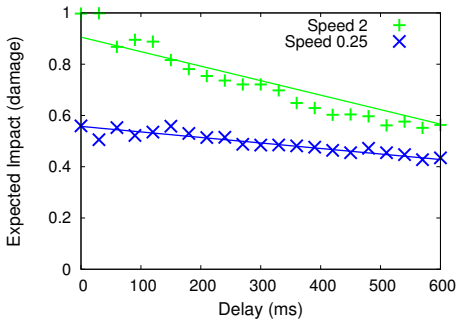

Figure 5: Expected Impact versus Speed

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 low precision AoE 7 weapon is higher than that of the higher precision AoE 3 weapon since a larger area of effect is likely to do more damage. Moreover, the slopes of both lines are nearly the same. Thus, any added delay to fire actions degrades the impact of each weapon by the same amount.

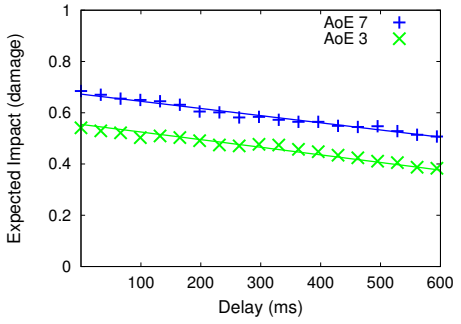

Figure 6: Expected Impact versus Area of Effect

6. MITIGATING LATENCY

This section describes how the precision, deadline & impact taxonomy (Section 3) and the results from Section 5 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 as to which action to transmit first. In order to minimize the impact of delay on player performance, the action 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. 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 action for W_R still in queue. The game has two choices: A) W_B can be placed after W_R in the queue, separated by a gap delay or, B) W_B can be placed before W_R, adding a gap to the delay for W_R. Since the goal is to maximize the total expected impact, the better choice, A) or B), depends upon which is larger: 1) expected impact of W_B at $R2$ + expected impact of W_R at $B1$, or 2) expected impact of W_B at $R1$ + expected impact of W_R at $B2$. Visually from Figure 2, the second option is considerably smaller than the first, thus W_B should be placed before W_R. Note that in a typical first-in, first-out (FIFO) system, the ordering would remain W_R before W_B, thus resulting in a larger degradation to player performance due to delay.

6.2 Experiments

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

In these experiments, 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 a role-playing or real-time strategy game is about 10 packets/second [3, 12] and the rate limit on uplink packets with the Valve Source game engine ranges from 10 to 100 packets/second. This suggests a gap between sequential packets of about 100 milliseconds. Thus, a minimum gap of 99 milliseconds (the equivalent 3 game loops) is used between packets for all experiments. A network latency of 66 milliseconds (the equivalent 2 game loops) is used for our 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 linear prediction for each weapon from the experiments in Section 5 in computing the expected impact.

For each queue type and each weapon configuration combination, Saucer Hunt was played 100 times, computing mean values of the total points with 95% confidence intervals.

Due to space constraints, results from AoE and speed are summarized only. 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 does not benefit from prioritizing actions based on AoE.

For Saucer Hunt, prioritizing player actions based on differences in weapon speed also does not have much benefit to player performance. Since the player fires slower weapons sooner than faster weapons, effectively "leading" the Saucer more, by the time the faster weapon is fired, the outgoing queue has cleared. In other words, when the second weapon is fired, there is no queue, so no opportunity for optimization. For games with a higher rate of player actions, especially those where the player takes several actions simultaneously, prioritizing player actions based on speed should show some benefit since weapons with different speeds have different slopes (see Section 5.3).

Figure 7 depicts the results for damage. The y axis is the player's performance, the total points, over a 90 second game. The x axis is the damage from bullets fired by the second weapon, from the same damage as the first weapon (1) to 16 times the damage as the first weapon. Each data point is the total points averaged over all 100 games run at that configuration, shown with 95% confidence intervals.[4] There are two trendlines – the blue '*' uses a FIFO queue and the green 'x' uses a priority queue where player actions that are separated by a gap are ordered so as to maximize the total expected impact.

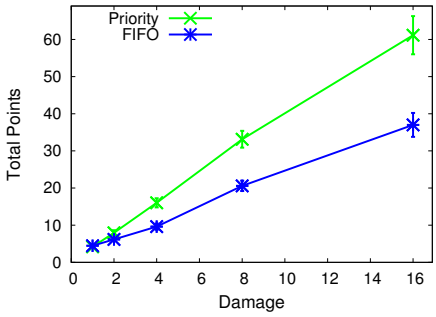

Figure 7: Player Performance

From Figure 7, prioritizing player actions based on differences in weapon damage can have a significant impact on player performance. This is explainable from Figure 4 that shows the expected impact of a weapon scales 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.

In general, 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 (e.g., Saucer

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

Hunt AoE), differentiating between them is not important – basically, player performance is the same no matter which action is transmitted first. When player actions have different slopes (e.g., Saucer Hunt damage), actions with a steeper slope are more sensitive to latency and should be favored.

7. DISCUSSION

The expected impact as measured for Saucer Hunt in Section 6 is the average of many actions. As for many games, the impact of any individual action can be tremendous ... or nothing at all, depending upon the state of the game when that action is performed. For example, in a side scroller, delaying a jump action when done on an empty platform may have no impact on player performance, but delaying a jump near a ledge may be disastrous. If the game client can determine the context in which the action is being performed it can adjust priorities accordingly. In the absence of this information, the average expected impact can be used.

Exhaustively measuring the impact of delay on all player actions is not feasible for most games. Ideally, the results presented quantifying the expected impact of delay on precision (area of effect), deadline (speed) and impact (damage) could be generalized for games beyond Saucer Hunt. While doing so is left as future work, the results from Saucer Hunt suggest a possible method that could be used for many games – 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 Figure 2; then 3) construct a linear equation from the line connecting the base and measured point, using this equation to compute the expected impact of that action in the presence of delay.

The relationship between player actions and latency is not linear in all cases. In fact, as suggested by [4] and data on Saucer Hunt with high speed weapons, actions that are the most sensitive degrade exponentially with latency. In such cases, an exponential curve likely fits better. However, for many actions over many latency ranges, a linear fit well-represents the degradation to performance due to delay. 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.

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 in network games is well-understood, the exact relationship between latency and different 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. The contributions include: 1) a taxonomy of player actions with 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 game systems with network bitrate constraints can mitigate the effects of latency on player performance.

For the game tested, a 2d arcade style shooting game, the expected impact of a player action 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 can show significant improvements to player performance.

The approach used in this paper was applied to a specific game, but the results may generalize to other games particularly those most similar to Saucer Hunt – i.e., arcade-style games with projectiles. Application to another game is left as future work, particularly games with different kinds of actions (e.g., moving an avatar or melee combat). In addition, developing a general model for the effects of delay on player actions without 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 *IEEE Int. Conf. on Networks (ICON)*, Sydney, Australia, Sept. 2003.

[2] 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.

[3] M. Claypool. The Effect of Latency on User Performance in Real-Time Strategy Games. *Elsevier Computer Networks*, 49(1), Sept. 2005.

[4] M. Claypool and K. Claypool. Latency and Player Actions in Online Games. *Comm. of the ACM*, Nov. 2006.

[5] M. Claypool and D. Finkel. The Effects of Latency on Player Performance in Cloud-based Games. In *NetGames*, Nagoya, Japan, Dec. 2014.

[6] 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.

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

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

[9] L. Pantel and L. C. Wolf. On the Impact of Delay on Real-Time Multiplayer Games. In *NOSSDAV*, Miami, FL, USA, May 2002.

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

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

[12] M. Suznjevic, O. Dobrijevic, and M. Matijasevic. MMORPG Player Actions: Network Performance, Session Patterns and Latency Requirements Analysis. *Springer Multimedia Tools and Applications*, 45, Oct. 2009.