

Latency Hiding through Player Behavioural Prediction in Highly Interactive Multi-Player Online Games

Chen Gao

Bachelor of Information Technology

Supervised by Haifeng Shen

Submitted to the School of Computer Science, Engineering, and Mathematics in the Faculty of Science and Engineering in partial fulfilment of the requirements for the degree of Master of Information Technology at Flinders University – Adelaide Australia

**Declaration**

I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is make in the text.

**Acknowledgment**

I wish to express my sincere thanks to **Dr. Haifeng Shen**, Senior Lecturer in School of Computer Science, Engineering, and Mathematics, Flinders University, for the continuous guidance.

**Abstract**

Online gaming industry has attracted a lot of attentions in recent years. With more and more players joining a game world, the designers of the game are facing increasing difficulty in addressing the issues caused by the varying quality of network connections. Although many approaches and methods have been developed to handle such issues, but players are still facing degraded game responsiveness or inconsistent game state. In this report, we propose a new approach that can predict a remote player’s behaviour using regression analysis when her/his network connection is lagging. To demonstrate and validate this approach, we have developed a 2D peer-to-peer table tennis game and collected a single user’s playing profile. The results show that the prediction accuracy is around 89%. We believe that the prediction accuracy can be improved by collecting more user profiles and further refining the prediction model.

*Keywords – Multi-player Online Game, MOG, Latency, Consistency, Responsiveness, Regression Analysis, Behavioural Prediction.*

Table of Contents

[1. Introduction 6](#_Toc422491490)

[1.1 Latency 6](#_Toc422491491)

[1.2 Responsiveness vs Consistency 7](#_Toc422491492)

[1.3 Objectives and Thesis Structure 8](#_Toc422491493)

[2. Network Architectures and Communication Protocol 9](#_Toc422491494)

[2.1 Network Architecture 9](#_Toc422491495)

[2.1.1 Server Centred Architecture 10](#_Toc422491496)

[2.1.2 Peer to Peer 11](#_Toc422491497)

[2.2 TCP and UDP 13](#_Toc422491498)

[3. Common Industrial Solutions 14](#_Toc422491499)

[3.1 Time Manipulation 14](#_Toc422491500)

[3.1.1 Time Delay 14](#_Toc422491501)

[3.1.2 Local-Lag 15](#_Toc422491502)

[3.1.3 Time Warp 15](#_Toc422491503)

[3.2 Prediction 16](#_Toc422491504)

[3.2.1 Dead Reckoning 17](#_Toc422491505)

[4. Player Prediction and Regression Analysis 20](#_Toc422491506)

[4.1 Prediction 20](#_Toc422491507)

[4.2 Regression Analysis 21](#_Toc422491508)

[5. Game Overview, Prediction Models and Results 23](#_Toc422491509)

[5.1 Game 23](#_Toc422491510)

[5.1.1 Latency Environment Simulation and Its Impact 24](#_Toc422491511)

[5.2 Prediction Models and Test Results 28](#_Toc422491512)

[6. Conclusion and Future Works 36](#_Toc422491513)

[7. Reference 37](#_Toc422491514)

[Appendix A: List of Future Works 39](#_Toc422491515)

[Appendix B: Regression Analysis of Each Model from Excel 40](#_Toc422491516)

# 1. Introduction

Multiplayer Online Game industry has been developed in a rapid pace and attracted a lot of attentions. Various types of games are published every month. People may play cooperatively against powerful monsters or compete with each other without considering geographic distance as the Internet can link people together in the world. However, the present internet technology limitations downgrade online games’ playability and one of the common issues is network latency.

## 1.1 Latency

The internet has a number of advantages, such as rich in various type of information or establishing communication without geographical constraints, but speed or stability are not included. The internet, or in other words, a Wide Area Network (WAN) is usually slower than other types of networks used to connect servers and clients across the globe through cables (Finn 2015). Transmitting data through optic fibre cables in WAN is just like a train system. When data are in transmitting, a large chunk of data is cut into small data packets which will be reconstructed back together on receivers’ side. Each data packet could be considered as a passenger on a train, and the train is the cable which buried underground or undersea, connecting to every building on the earth. The fastest train Shanghai Maglev can reach 430 kilometres per hour, so for a 4300 kilometres nonstop journey, passengers need to spend at least ten hours on the train (Shanghai Airport Authority 2015). Same to networks, data are coded in light and transmitted in optical fibre cables from source address to hops which act as interchange, then finally reach their destination, light can travel 1.07925285 × 109 kilometres per second in void, but in fibre or copper cable, it slows down to about 60% of its normal speed (Hargreaves N.D.). Thus if someone wants to send a message from Seattle to England which is 7730 kilometres away, then it takes about 26 milliseconds, round trip is 52 milliseconds. That 52 milliseconds, the time of this process is so-called latency - how long does it take for data/messages to travel from source address to its destination and the destination confirms message is received back to source, often known as Round Trip Time (RTT).

Many factors can have effect on this process, some are structural, such as number of hops, or in other words, routing path certainly can decide the “distance” data is going to travel, implementing different routing algorithms may result in completely different outcomes of network latency. Some others are not, like hop response time which also is an important factor. The internet is unpredictable, if one of hop on routing path is too busy to deal with large amount of exchange requests when it receives our data, then it is not very difficult to imagine that the data arrival will be delayed by that hop. Doctor Moon (2007) listed a table of all possible factors that could result in network latency which include TCP layer buffering, thread priorities, speed of light, store and forward routers, over-used router and number of routers.

## 1.2 Responsiveness vs Consistency

Responsiveness and consistency are two important factors to evaluate a Distributed Interactive Application (DIA), such as online games. Consistency refers to that each node on network has the same data state. Responsiveness refers to the delay for an update register throughout network (Smed & Hakonen 2006). Consistency and responsiveness are opposite, people may have experienced situations like seeing dead man shooting in game world. it’s a classic example that local game state determines a man was shot and died, but on server side, the man is still alive due to the message that indicates the man’s death hasn’t reached server yet, thus although the game is very responsive, the player is misguided by wrong information. The contrary example is a player fires a weapon and the visual effect of the firing can be observed with a short delay. This is due to the message of firing the weapon need to reach server first, after server confirms the fire, message travels back to player machine, then displays visual effect. The consistency check on firing the weapon makes game feel sluggish, or in other words, irresponsive.

Traditionally, responsiveness is considered less important than consistency, however, in some real-time tasks such as online gaming, responsiveness becomes a very important factor to playability, in highly interactive online games, such as sport games, require stable networks which no one can guarantee, and the responsiveness requirements for these types of game are even higher than other types. At present, it is still impossible to eliminate network latency, thus, various consistency models were developed trying to find the balance between consistency and responsiveness (Moon 2007), some of models are being too optimistic to guess game states which may have heavy effect on consistency, others are being pessimistic that makes game feel slow.

## 1.3 Objectives and Thesis Structure

The purpose of this research to review current methods of dealing latency and purpose a new way to avoid network limitations and address following situation:

***While players encounter a sudden lag spike during playing, is there a way to keep game running smoothly without player noticing the lag.***

The rest of paper is organised as follow, in section 2, we explore how network architectures and communication protocols may have influence on latency. Then in section 3, we review what common industrial solutions are to deal with network latency. In section 4, we introduce our theory of dealing latency, methodology and in section 5 explain game design and experiment results. Finally in section 6, we will conclude our current work and discuss the future work of this project.

# 2. Network Architectures and Communication Protocol

Different network architectures and communication protocols have different specialties, some architectures and protocols may perform well in one type of game, but underperformed in some other types. Choosing the right architecture and protocol is significant to game design, in this section, we explore the differences of common network architectures and communication protocols, introduce any latency handling technique specifically targeting them.

## 2.1 Network Architecture

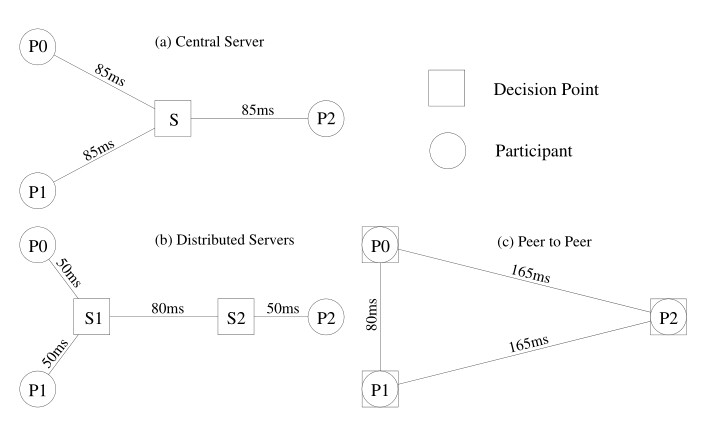
In modern game industry, there are three types of network architectures commonly used in Multiplayer Online Games (MOGs) which are Client/Server(C/S), Peer to Peer (P2P) network and Distributed Servers. Distributed Servers architecture shares a lot of features with Client/Server, thus, we are going to focus on two main types: C/S and P2P.

Figure 2.1.1 Common Online Gaming Network Architectures (Brun et al. 2006)

### 2.1.1 Server Centred Architecture

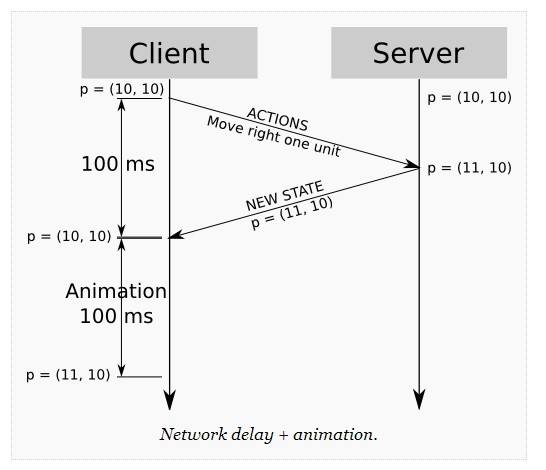
Client/Server (C/S) is the most widely used architecture in game industry field. In this architecture, clients do not directly communicate with each other, all messages must be send to server first, and then server relays messages to other clients. Due to server maintaining game state and all clients only read game state from server, game consistency can be well maintained. However, all server centred architectures (C/S and Distributed Servers) share similar situations that all data need to be processed by server, even if a local player tries to make their own game avatar performs a basic action like move forward one step, the player still needs to wait until server confirms this movement and acknowledge the local game program, then local player can see the avatar in game moved. As below figure 2.1.1.1 showing, assuming all messages need 50 milliseconds to travel to server and server acknowledgement takes same time to get back. When local player make an input cause game avatar position move from (10, 10) to (11, 10). In 100 milliseconds later, after a short delay, local player sees the avatar start to move (Gambetta N.D.).

Figure 2.1.1.1 Local Move Waits Server Confirmation (Gambetta N.D.)

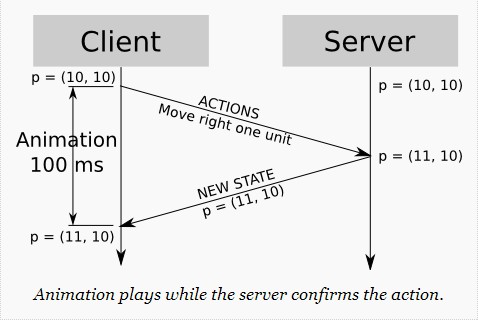
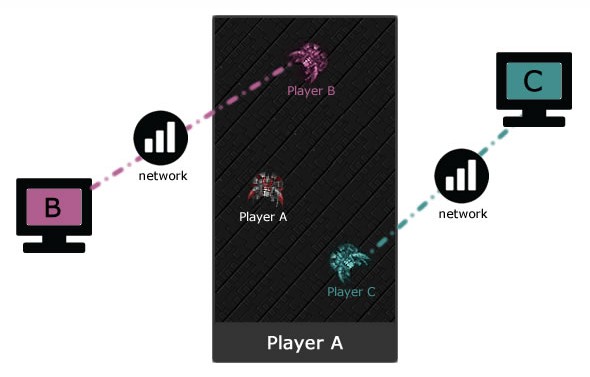
A common way to work this around is client side prediction. Instead of waiting server confirms the movement, we can assume that server will accept the new game state and game program can move the game avatar before server confirms it because we know exactly what would happen. Even if server denies this movement, we can later on rollback the game avatar to previous state. Rollback operations may cause confusion, but in most of time, client side prediction is used when we know server will confirm the movement very soon. Thus, local player can see game avatar moved immediately after player press a key, but the message sent to server is still travelling on the Internet.

Figure 2.1.1.2 Client Side Prediction (Gambetta N.D.).

### 2.1.2 Peer to Peer

Most of early stage networked games and army simulation systems are designed in Peer to Peer (P2P) architecture. Differ from Client Server architecture, P2P does not have an central entity acts as a server, all participants have equal privilege and directly communicate with other peers. In a P2P game, every peer has their own game state, if one peer makes a change on a local object, then the peer updates the object state to others. Because of central entity absence, there is no true game state, maintaining consistency in a P2P game is much harder than a Client Server game, but the network latency is also lower than C/S architecture as there is no need to relay messages to server to check messages’ validity (Fiedler 2013).

Figure 2.1.2.1 Peer to Peer Networked Game Example

A typical way to synchronise game state in P2P architecture is called Lock Step which was wildly used in early stage P2P games. Its basic idea is all peers have a synchronised clock, if one peer makes a change, the peer sends update to all other peers along with incremented clock. No one is allowed to change game state until all peers have confirm the updated clock, thus before every round of changing game state, Lock Step ensures all peers have the same game state, everyone knows what others have done in the current round, then next round can start. The downside is the time of each round is limited by the latency, because in each round every peer needs to receive update from all other peers, if one of them has unusual high latency, then the time of each round gets longer to wait that peer. It may be still good if the game is turn based, that is also the reason why modern Real Time Strategy (RTS) games abandoned P2P network.

Not like Client Server architecture, there is no way to improve P2P latency issue that specifically targeting P2P network. All local operations are effective immediately on screen, there is no need of client side prediction.

## 2.2 TCP and UDP

All communications between peers or between server and client involve communication protocols: TCP and UDP. The biggest feature of Transmission Control Protocol (TCP) is that it keeps tracking the data that has been sent to ensure the data arrive at their destinations in order. TCP sender expects to receive an acknowledgement (ACK) from receiver for every packet has been sent. If sender do not receive the expected acknowledgement within a timeframe, then sender will stop transmitting other data packets and re-send the unacknowledged packet. Some Sliding Window Protocols may improve this situation by allowing sender to keep sending packets even pervious packets haven’t been acknowledged. But once the number of unacknowledged packets reaches protocol threshold, eventually sender has to stop transmitting new packets, send one packet for one acknowledgement packet received (Lam 2014). Thus, to ensure its reliability, TCP needs to wait for any delayed or lost packets until the right packet comes in, then the message can be processed which makes TCP based online games suffer higher latency than others as the protocol sacrifices transmission performance to achieve high reliability. As Fiedler (2013) stated in his article: “Using TCP is the worst possible mistake you can make when developing a networked action game like an FPS!”

On the other hand, User Datagram Protocol (UDP) is a connectionless protocol, it cannot ensure data will be received or the data order will be received in sequence, but its transmission rate is better than TCP as it does not require acknowledgements from receiver. Although UDP is preferred when designing an online game, it is still not possible to eliminate transmission errors that causing network latency. The time between sending a packet and realising that packet has an error during transmission could take few seconds, standard UDP would just discard any packet with error which cause “gap” on data, other customized UDPs may have error recovery mechanism, but this process certainly results in unusual high latency. The latency may not cause serious consequences if we are downloading a file or browsing a website. But for online gaming or any other real time systems, the result could be fatal (Fiedler 2013).

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| Packet Delivery Check | Yes | No |
| Delivery Order | First In First Out | No Guarantee |
| Transmission Rate | Slow | Fast |

Table 2.1 UDP and TCP

# 3. Common Industrial Solutions

Except those methods specifically targeting a network architecture or communication protocol, many common methods have been developed to improve latency issues. All methods can be classified into two types: time manipulation and prediction (Armitage et al. 2006).

## 3.1 Time Manipulation

Time manipulation methods work by buffering recent game state, and player always play under slight constant delay or execute player command in a past game state. There are many different methods to achieve time manipulation, such as time delay, local-lag and time warp.

### 3.1.1 Time Delay

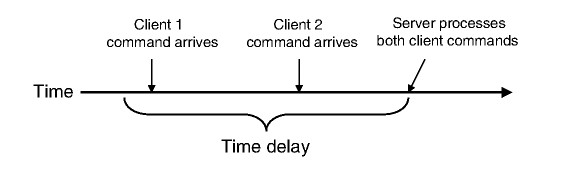
 A common method to deal with different latency between players is to delay operation on server side. Instead of executing user commands immediately, user commands are hold by server for a short period of time (Armitage et al. 2006). This process allows server to accurately compute results for players to play under different latency. As figure below shows, if client 1 is attacking on client 2 and at the same time client 2 is trying to dodge the attack. Client 1 has lower latency, client 1’s command arrives at server early. In normal setting, server would just execute the command. But client 2 dodged this attack at the same time, the command arrives late due to higher latency which results in client 2 got attacked that supposed to be dodged. With time delay, server waits a period of time after received client 1’s command, it is likely that client 2’s command will arrive shortly to against it, after a short wait, if client 2’s command arrived, then server can correctly compute the result that client 2 dodged client 1’s attack. In other words, time delay help to improve game fairness.

Figure 3.1.1.1 Server Delays Execution (Armitage et al. 2006)

However, time delay itself is also a type of latency - a player can see the change of game state after transmission round trip time plus execution delay. That is a trade-off time designers need to make, it sacrifices game responsiveness to improve game fairness.

### 3.1.2 Local-Lag

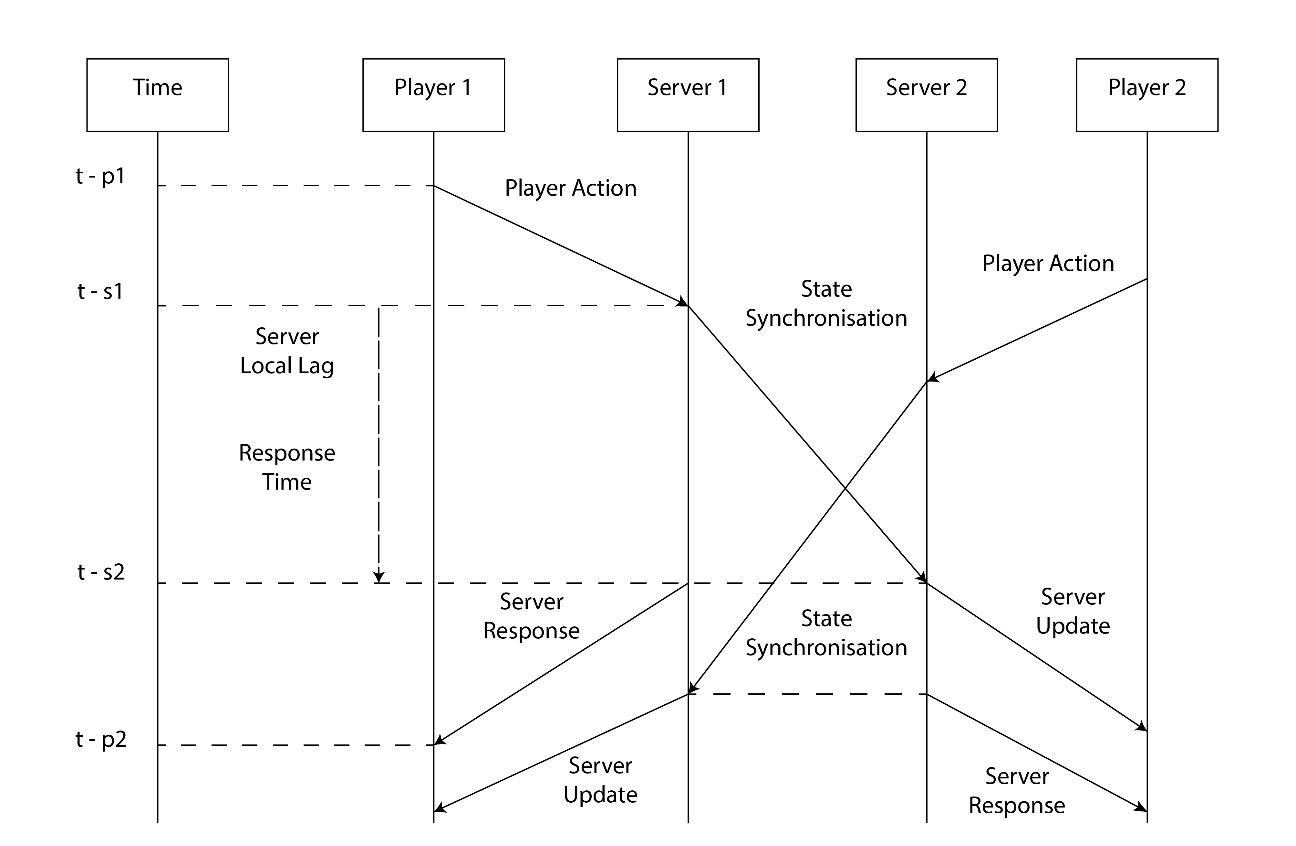
Local-Lag is another type of time manipulation method used in Distributed Servers Architecture. The idea is if two players play together through two different servers and one server received a client command, the server first send the command to the other server, then waits until the other server gets the command, both server process the command and send the result back to two players at the same time. As figure below shows. Player 1 (P1) first arrives at server 1 (S1) at , but the game state on S1 remains until server 2 (S2) receives the command, then the result be sent back to P1 at . In this way, Local-Lag method sacrifices game responsiveness, in exchange, it keeps the consistency between server 1 and server 2 (Liang et al. 2006).

Figure 3.1.2.1 Local-Lag causes P1 command to wait until S2 receives the command.

### 3.1.3 Time Warp

The most wildly used time manipulation method is time warp method, or in other words, rollback method. With time warp, server keeps some snapshots of every players’ recent game state. If one player’s command involves another player, server calculates both players’ latency, rollbacks back game state to the pervious state of the exact moment player sent the command (Armitage et al. 2006). The general algorithm on server side is:

1. Receive packet from player 1
2. Interpret packet content
3. Calculate elapsed time = current time - client 1 ping
4. Rollback player 2 game state based on elapsed time
5. Execute player 1 command
6. Repeat states from rollback state to current state

A good commercial example of time warp is Half Life 2 developed by Valve on Source Engine. Screenshot below was taken on a server while client has 200 milliseconds latency. The red boxed model was the client side interpolation (explained later) 100 milliseconds ago, and the blue boxed model was the player’s position before command execution. Thus, when server received fire command from the shooter, server restored target client state to blue boxed position, and confirmed the hit, even the aim on screen was completely off the target (VALVE 2014).

Figure 3.1.3.1 Time Warp Checks Player Recent Game States

## 3.2 Prediction

Time manipulation methods are good at maintaining consistency between server and client, but it reduces game responsiveness. Predictions, on the contrary, is another type of latency handling technique which intends to minimise the impact on game responsiveness. Client side prediction we mentioned in Client/Server section is one form of these methods which predicts server response to local player commands.

In this section, we will focus on opponent prediction – predicting what other players are doing (Armitage et al. 2006).

### 3.2.1 Dead Reckoning

Dead reckoning (DR) was firstly used in navigation field, later on became a method to deal with latency on network gaming. Its idea is simple, predicting a player or an object future state based on its current state. The prediction assumes the target path is ballistic, thus it calculates target future position by physical laws.

C:\Users\cst\Desktop\thesis pics\vel0.jpg Assuming target has zero velocity:

C:\Users\cst\Desktop\thesis pics\vel1.jpgAssuming target has constant velocity:

C:\Users\cst\Desktop\thesis pics\vel2.jpgAssuming target has constant acceleration:

There are nine standard algorithms for dead reckoning, some others are repeated due to use of different coordinate system, and above three are the mostly used (Liang 2007). Dead reckoning, in fact, is more focusing on eliminate bandwidth issues rather than latency issues. It defines a threshold on local player track, if local player is moving within the threshold, then dead reckoning allows remote players predict local player’s state based on last update. Only when local player reaches or moves out of the threshold, then dead reckoning sends a new update to other players, indicates local player current state. Thus, to get any new update, remote player still need to wait messages travel through network which messages may delayed.

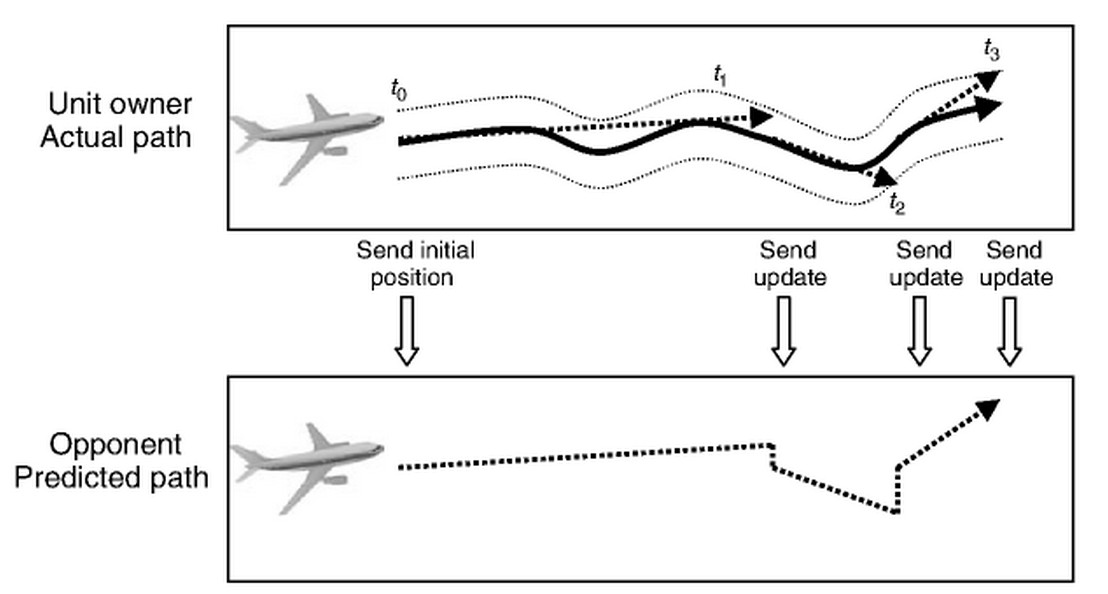
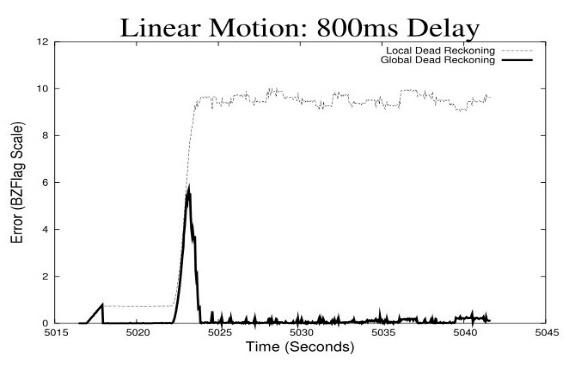
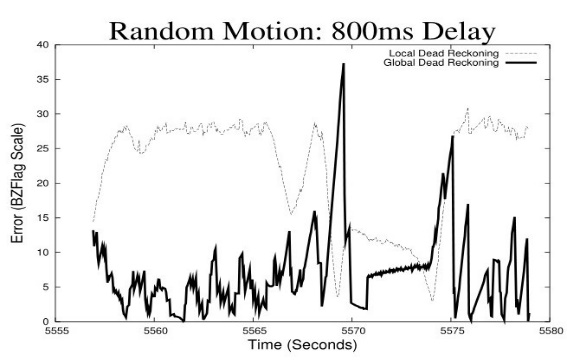
Figure 3.2.1.1 below shows an example of Dead Reckoning, the unit owner only sent update to others when the jet reached the threshold which were at , and . Although the owner also changed heading between and , but the track was still within threshold, thus no update was sent to opponent player. Because of this feature, dead reckoning requires less network communication than other algorithms, thus it requires less bandwidth. But even players are playing under a perfect network condition (eg, less than 50ms), players still only get updated until the opponent goes off the track too much which introducing a new factor to cause latency.

Figure3.2.1.1 Example of Dead Reckoning Prediction and Update Rate (Armitage et al. 2006)

Although dead reckoning may improve game responsiveness in high latency environment, but it comes at a cost of reducing consistency. Its predictions assume all objects has a ballistic track, but in reality, a player’s movement is hardly to be linear. A research done by Aggarwal, Banavar and Khandelwal (2004) shown in BZFlag game under 800 milliseconds latency, there are huge differences on dead reckoning results when player making linear movement and making random movement. That is dead reckoning’s weakness, it cannot predict changes on heading or velocity.

Figure 3.2.1.2 Larger Error Rate When Making Random Motion (DR)

Another method is often mentioned with prediction, or in other words extrapolation, is called interpolation. They are both used on client side, but not like extrapolation which predicting game objects future state, interpolation tends to slow down the process of client synchronising with server. Interpolation allows client creates an intermediate state between current local state and received server state to make games run smoother, avoiding sudden jump on game objects which is similar to time manipulation methods.

It is not hard to understand that time manipulation methods tend to downgrade game responsiveness to maintain consistency, and prediction in some degree can maintain game responsiveness but downgrades consistency for it predicts object tracks rather than what player is doing. In next section, we’ll talk about our theory and method that predict what players are doing to hide network latency.

# 4. Player Prediction and Regression Analysis

Behavioural analysis are commonly used to design Artificial Intelligence (AI) in game industrial fields, some AI have ability to predict players’ next move and work out counter moves. In this section, we introduce our theory and some similar researches have done in the same field.

## 4.1 Prediction

Behavioural science believes that every individual has their unique marks, it might be represented in habits or catch phrases (Wright 2000). In game, it’s the same. Some players often turn right to pass barriers in game world, others are used to turn left. Player modelling and player profiling are not new concepts to game researchers, most of level designs in games are relate to these two theories which could make game AI has similar style and skills to cooperate players or against players and keeping game difficulty at an appropriate level (Bakkes et al. 2012). Van den Herik et al. (2005) discovered that in 1970s, when against a stronger player, chess game AI may accept draw while in advantage, but deny draw in disadvantage while against weaker player. Some AI can even predicts unreasonable moves. With further researches, we can predict not just players’ decisions but also actions which can be applied to some fast paced games, like first-person shooting and sport games.

Despite that there is still no game successfully implemented remote player prediction to handle network latency issues, but there are many people researching in relative fields. The most likely successful research is Microsoft DoLorean project. In 2014, researchers in University of Michigan, Microsoft Research Division and Siberian Federal University (Lee, et al.) published a research paper claimed their speculative execution system Outatime can mask up to 250ms latency for cloud gaming. The Outatime was built on Markov Chain theory which can predict objects future state based on current time and state (Keilson & SpringerLink 1979). Given players’ historical tendencies and recent performance, Outatime can render next frame ahead entire RTT when latency is lower than 250ms. Except Microsoft, there are many other theories, like Becht and Bakkes’s meIRL-BC theory (2013) - predicting player positions based on Maximum Entropy which is considered the most objective way to predict a random event by maximise the uncertainty of random variables (Wu & SpringerLink 1997). And his later theory improved meIRL by using general behaviour classification which was built on Random Forest theory that creates multiple decision trees and all cast a vote when classification is needed (Becht 2013). Some other researches treated future events like consequences of a serial actions rather than random cases, for example, Harrison and Roberts’s research considered that actions in game are sequential, they tried to predict players’ next move based on what players have done (Harrison & Roberts 2011).

Traditionally, player models and profiles are created to tell game AI how to play against players, AI knows how players usually win, so it can avoid similar situations and make games more challenging (Webersdorfer 2012). However, if we can find all factors that decide players’ every action, then we can profile the player and use the profile to predict what the player is doing during network lagging. Our idea equivalents to that let players play a local, non-networked game during lagging and the game AI tries to imitate the opponent player.

## 4.2 Regression Analysis

The method we used to model players is called regression analysis which is a statistical way to estimate relationships between variables. Change of some variables, for example, Product Unit Price may have influence on Product Sold Amount. Regression analysis can reveal such relationships, describe relationships in mathematical expressions, and indicate the degree of dependence between the independent variables and the dependent variable (Sykes N.D.), it’s been wildly used in market forecasting. Least Squares Method is the standard approach of regression analysis. Ordinary Least Squares (OLS) assumes that there is a linear relationship between the independent variable X and the dependent variable Y, and the relationship is expressed as: Y= aX+ b + e;

Where:

X and Y are observed data (shows as blue dot in chart next page)

a and b are estimated by regression (slope and intercept)

e represents standard error, other factors may also have influence on Y

Figure 4.2.1 Example Regression from Excel

In reality, as figure 4.2.1 above shows, it is very rare that a relationship between two variables perfectly matches a mathematical expression which means it’s impossible to find a line that strictly fits all data, therefore, Least Squares Method aims to find the next best line that fits most of data, by calculating the sum of squares for each point’s error, looking for the line has minimal error to all points.

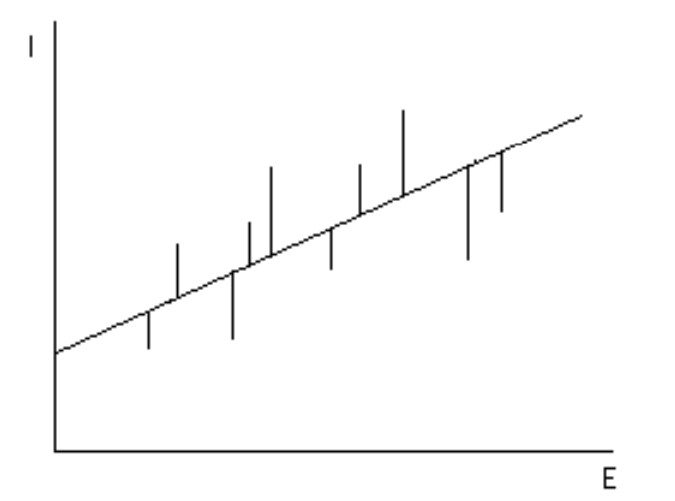
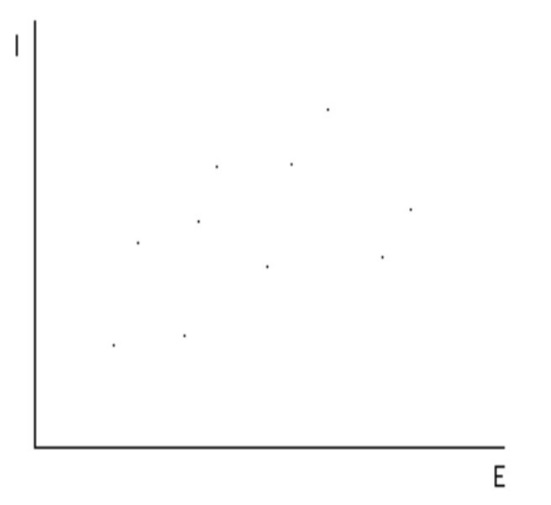


Figure 4.2.2 Least Squares [5] – Minimal Error to All Points

# 5. Game Overview, Prediction Models and Results

## 5.1 Game

To verify our prediction theory, a peer to peer ball game was developed. In our game prototype, two players, each controls a paddle on top or bottom of screen (local player’s paddle is always at bottom of screen) with a window size of height 800 and width 580. Players need to keep bouncing a ball on screen back to the opponent. If a player fail to bounce the ball back, then game is over.

Game updates all data 60 times per second (FPS 60). Paddles can only be moved horizontally - to left or to right, and paddle speed is constantly 10 per frame - (±10, 0). Ball has an initial speed of (5, 10) when game starts, and its horizontal speed slowly accelerate by (±2, 0) based on how many times the ball has been bounced, every time the ball reaches left or right window border, the ball reverses its horizontal direction.

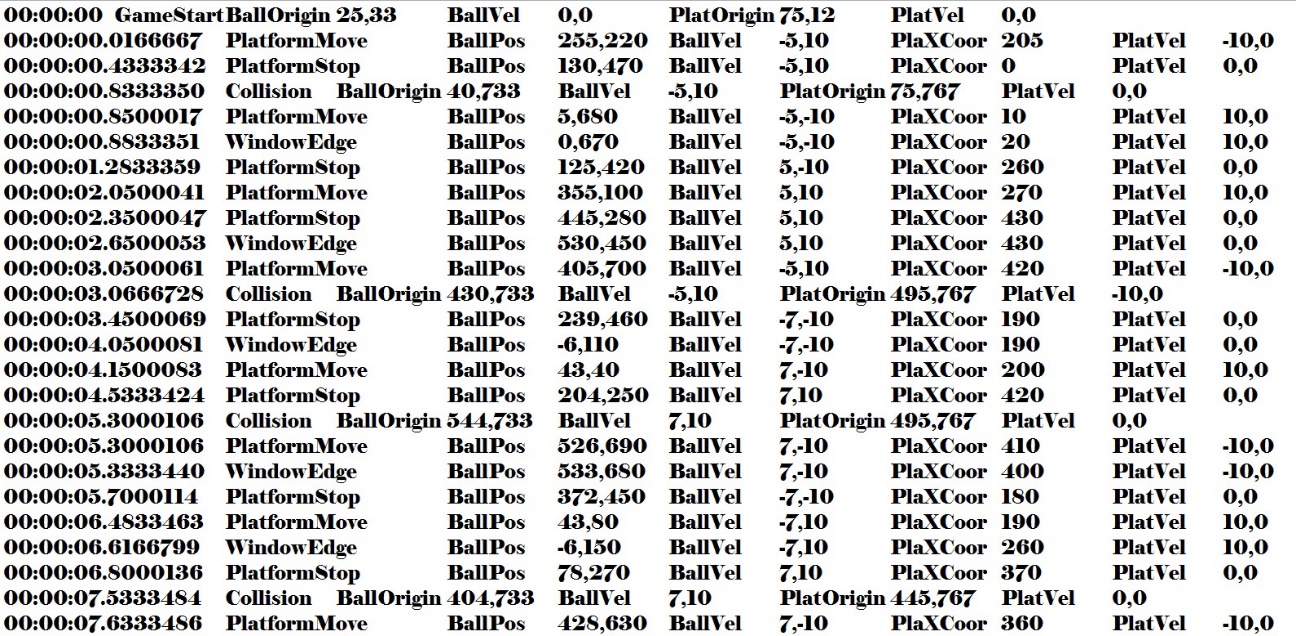
A single player mode and a multiplayer mode are available. In single player mode, player will play against a prefect computer player, and local player’s performance data are collected during play, including when player starts to move paddle, when stops paddle, when the ball collides with a paddle along with ball position, ball velocity, player paddle position and player paddle velocity.

Figure 5.1 Sample Data Set

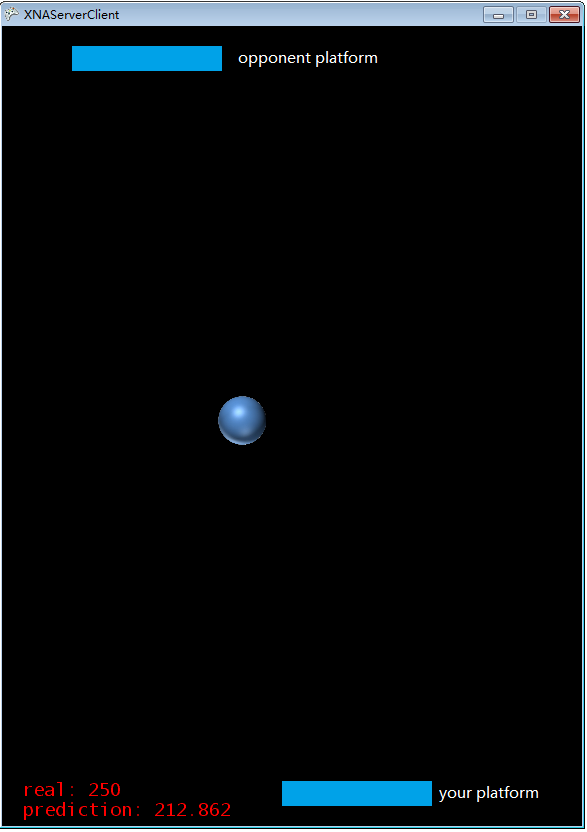


Figure 5.2 Single Player Mode

### 5.1.1 Latency Environment Simulation and Its Impact

Multiplayer mode is built on XNA 4.0 refresh/XBOX Live service which allows two players under same subnet play together. XNA framework allows programmers test online games without connect to the Internet. One of game session property – session.simulatedLatency can simulate lagging environment on a LAN game. With a random number generator, it is possible to create an unstable, unpredictable network connection on LAN. With a counter, we also can control how often to generate latency.

Random rnd = new Random();

Int lag = rnd,Next(1, 1000); //creates a number between 1 and 999

Timespan time = new Timespan(lag);//define timespan based on random number

session.simulatedLatency = time; //generate latency around the number

A high network latency would cause many strange phenomena due to inconsistency. In tests, we set latency to 800 milliseconds which is a very high for online games. Initially the latency causes slow reaction on paddle movement, the ball and the remote paddle sometimes jumped back a short distance during movement. This situation is because every time player A received an update from player B, the ball and the remote paddle were re-placed to a past position (around 800ms ago in this case). If two players keep updating their game state to the other side, then the ball and the remote paddle will keep jumping back to where they were.

Later on, we implemented a basic dead reckoning algorithm, all objects moved smoother than they used to be, but the remote paddle always jumped back a short distance when it stopped. This phenomenon is caused by dead reckoning itself because game program cannot foreseen when player stops moving the paddle until the exact moment player releases keys, and dead reckoning method assumes the player kept moving the paddle until the stop message came in which the message was delayed by network latency.

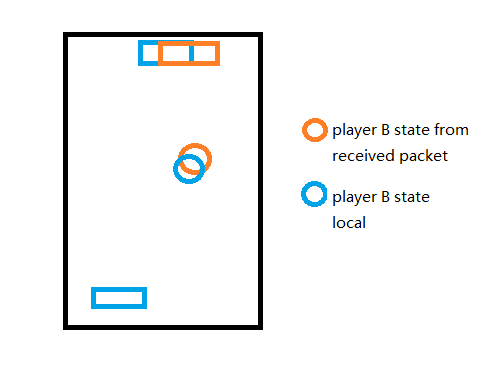


Figure 5.1.1 Latency Causing Game Objects Jumping

In our reworked multiplayer mode, instead of using dead reckoning, we tried to make game program imitates remote player while network is lagging.

Before game starts, if two player are matched up, a performance file should be send to the remote side, it contains few equations used to predict the local player’s move, including when to move paddle and when to stop paddle. After player profile exchange, game can starts like a normal Peer to Peer game, all players control their paddle on local, and keep updating local paddle position to remote side along with ball position and ball velocity.

If network gets lagging during game, host first sends a ‘network lag’ flag to the other player, then host game program ignores all incoming packets from the remote side, takes control of remote side paddle, moves and stops the paddle based on prediction models. The other side does the same once receive ‘network lag’ flag from host. At this moment, both game host and client are all playing a local game which the game AI is trying to imitate the remote side player.

Once network returns to normal, host sends a ‘network restore flag’ to the other player, when client side receives that flag from host, client side immediately sends a packet contains current game state. Host then compares local game state with client’s state, tries to synchronize two sides. After game state synchronization is finished, host and client re-start to process incoming packets, game returns to normal.

Maintaining the consistency between two game states is critical to this method. As long as prediction errors are not large enough to cause player miss the ball, then there still is a possibility that we can fix the inconsistency without player even notice it. On the other hand, even most of time prediction errors are in safe range to allow us keep maintaining the consistency, a single large error may cause program predicts the remote player is going to miss the ball, but in fact the remote player successfully bounced the ball back, and local player sees the ball mysteriously bounced back which the remote paddle clearly missed it. Due to the time limitation, at current stage, we only focused on improving the accuracy of our predictions, the more accurate our predictions are, the less inconsistency we need to fix, consistency maintenance and recovery issues will be explored in later stage.

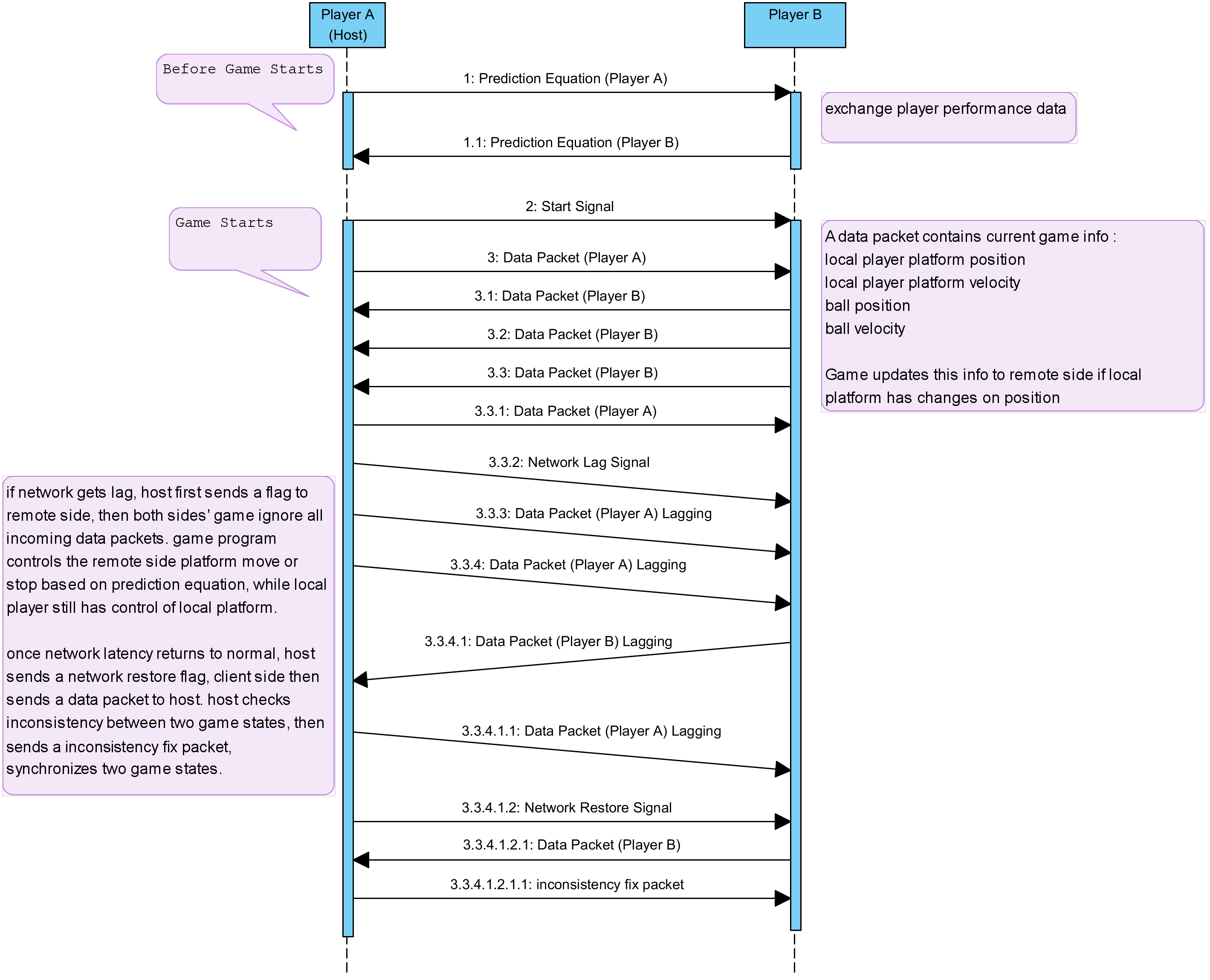


Figure 5.1.2 Example Game Communications

## 5.2 Prediction Models and Test Results

To create a regression model, we collected 18 games data of a single test subject and we proposed our first model used to predict when the player moves the paddle which predictions are based on the relationship between the distance from the ball to the player paddle and ball’s vertical coordinate when player starts to move the paddle. Regression analysis shown a strong relationship between these two variables (Multiple R = 0.969575648), and 94% (R Square) of data we collected fit to this model. Both independent variable P-value and regression Significance F are less than 0.05. Standard Error which indicates prediction accuracy was 29.1, this value shall be compared with other models’ standard error to determine which model has a better accuracy (the lower the better) (Palmer & O’Connell 2009).

Explanation of each result (Wilcox 2010)

Multiple R: Correlation coefficient, a value of 1 means perfect relationship, a value of 0 means no relationship at all.

R Square: Coefficient of determination, indicating how many data fit the model

Adjusted R Square: adjust for number of terms in model, replace R Square when having more than one independent variable X.

P-value: Hypothesis test, a value larger than 0.05 means the variable has little value to the model.

Significance F: Probability that equation does not explain the variation in dependent variable Y, a value larger than 0.1 indicates non-meaningful correlation.

|  |  |  |
| --- | --- | --- |
| Model No. | 1st | |
| Regression Result | Value | Comment |
| Multiple R | 0.9696 | Strong Relationship |
| R Square | 0.94 | Data Matched |
| Adjusted R Square | 0.9397 | Not Used |
| Significance F | 3.7134E-102 | Meaningful Correlation |
| Standard Error | 29.1367 | Compare to others |
| P-value | < 0.05 | All variables |

Table 5.2.1 Regression Analysis of the1st Model

Complete Regression Analysis could be found in Appendix B

Formula used for predictions (1st):

*Ball vertical coordinate = Distance (ball to paddle) \* -0.989707185 + 783.8653914*

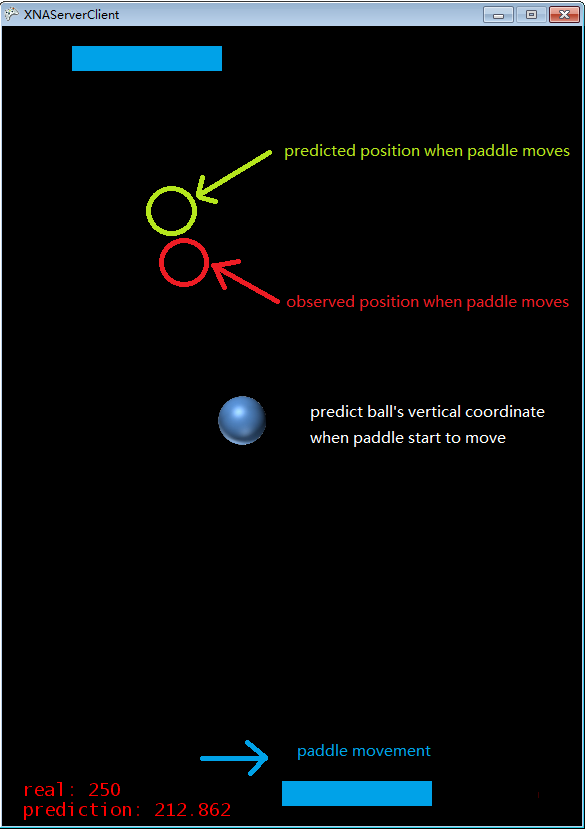


Figure 5.2.1 Example Prediction

The experiments was executed 3 random days in a week, 3 to 4 games each day, 11 games in total were recorded for the 1st model. 174 predictions were successfully produced out of 193 attempts. Because the casual relationship was selected for this model, we sometimes got several prediction results which all fit the model, in those cases, only the first prediction was selected as the final result. Prediction results shown an average ±171.9 error from real observed value, maximum error -599.324, and minimum error 1.9. Considering the ball’s vertical speed was 10 per frame, game’s frame per second (FPS) was 60, 171.9 error needs to take 17.19 frames and the delay was 17.19/60 which can be considered as ±286.5 milliseconds latency. However this model was determined unusable for the maximum error 599.324 is very close to the distance between two paddles which is 700 and a large number of prediction errors are in [-200, -400].

Formula Used to Convert Mean Residual to Latency:

*Latency = (Residual / Ball Vertical Speed) / FPS*

The residual result for the 1st model shown the error of each prediction in experiments. A negative value means player moved the paddle early than predicted, a positive value means player moved the paddle after we predicted. It is unlikely that the distance between the ball and the paddle is the only factor causes player move the paddle, but this model could be used as a baseline to examine our future models.

Figure 5.2.2 Residuals 1st Model

In the 2nd model, we took ball velocity into consideration which slightly improved R Square to 0.94311 and standard error reduced to 28.3. We were able to predict 230 results out of 236 attempts. The P-value of new introduced variable ball velocity is 0.0012 (less than 0.05) and all others P-value and regression significance F remains in safe range.

|  |  |  |
| --- | --- | --- |
| Model No. | 2nd | |
| Regression Result | Value | Comment |
| Multiple R | 0.9715 | Strong Relationship |
| R Square | 0.9438 | Not Used |
| Adjusted R Square | 0.9431 | Data Matched |
| Standard Error | 28.303 | Lower than 1st |
| Significance F | 1.2659E-102 | Meaningful Correlation |
| P-value | < 0.05 | All variables |

Table 5.2.2 Regression Analysis of the 2nd Model

Formula Used for Predictions (2nd):

*Ball vertical coordinate = Distance (ball to paddle) \* -0.961346383 + Ball Velocity \* 2.661065859 + 728.9963297*

The experiments for the 2nd model was executed in similar conditions, 14 games was recorded. The result shown an average error of ±143.43, the maximum error was -531.564 and the minimum error was 0.001. Given the calculation from last model, the average ±143.43 error could be considered as ±239.05 milliseconds latency.

Figure 5.2.3 Residuals 2nd Model

In our 3rd model, we further took two more variables into consideration which were the ball’s horizontal coordinate and the paddle’s horizontal coordinate. The Adjusted R Square shown improvement from 0.943 to 0.947 and standard error was 27.1. The P-value for new variables are 0.001 and 0.003.

|  |  |  |
| --- | --- | --- |
| Model No. | 3rd | |
| Regression Result | Value | Comment |
| Multiple R | 0.9741 | Strong Relationship |
| R Square | 0.9489 | Not Used |
| Adjusted R Square | 0.9476 | Data Matched |
| Standard Error | 27.1563 | Lower than 2nd |
| Significance F | 8.2716E-103 | Meaningful Correlation |
| P-value | < 0.05 | All variables |

Table 5.2.3 Regression Analysis of the 3rd Model

Formula Used for Predictions (3rd):

*Ball vertical coordinate = Distance (ball to paddle) \* -0.965004073 + Ball Velocity \* 2.240853224 + Ball Horizontal Coordinate \* 0.041043525 + Paddle Horizontal Coordinate \* 0.102167761 + 706.3240293*

The experiments results shown two new variables greatly improved prediction accuracy from average ± 143.18 to ±100.76, the maximum error was -449 and the minimum error was -1. The average ±100.76 error could be considered as ±167.93 milliseconds latency.

Figure 5.2.4 Residuals 3rd Model

Based on test subject’s game experience, we analysed any potential factors could make a player performs better and we proposed a new idea that the ball position where it reaches window border and reverses its horizontal direction might be significant to the timing the player moves the paddle. Thus, we recollected 18 games data and applied new data to previous models and compared with the 4th model. However, the regression analysis shown that the new introduced variable resulted in a great decrease on Ball Velocity’s P-value which was 0.465(larger than 0.05), therefore, the velocity variable was taken out from the new model.

|  |  |  |
| --- | --- | --- |
| Model No. | 4th discarded | |
| Regression Result | Value | Comment |
| Multiple R | 0.9748 | Strong Relationship |
| R Square | 0.9503 | Not Used |
| Adjusted R Square | 0.9478 | Data Matched |
| Standard Error | 26.0299 | Lower than 3rd |
| Significance F | 7.23735E-63 | Meaningful Correlation |
| P-value | 0.465 | Ball Velocity |

|  |  |  |
| --- | --- | --- |
| Model No. | 4th (no ball velocity) | |
| Regression Result | Value | Comment |
| Multiple R | 0.9747 | Strong Relationship |
| R Square | 0.95 | Not Used |
| Adjusted R Square | 0.948 | Improved from discarded |
| Standard Error | 25.9697 | Lower than discarded |
| Significance F | 4.06494E-64 | Meaningful Correlation |
| P-value | < 0.05 | All Variables |

Table 5.2.4 and 5.2.4.d

Regression Analysis of the 4th Model and the discarded Model

Formula Used for Predictions (4th):

*Ball vertical coordinate = Distance (ball to paddle) \* -0.950357277 + Ball Horizontal Coordinate \* 0.049730383 + Paddle Horizontal Coordinate \* 0.165261655 + Last Ball Vertical Coordinate at Window Border \* 0.065508861 + 706.5644181*

After taken out ball velocity variable, we got an analysis of Adjusted R Square 0.948 and standard error 25.9, both were better than previous models. The experiment result shown an average error of ± 80.9894, the maximum error was 321 and the minimum error was 0. The average error can be considered as ±134.98 milliseconds latency. However, due to the strict conditions were used for improvement on accuracy, we were only able to successfully predict 190 results out of 270 attempts. From the figure 5.2.5 below, we can see that most of predictions were within [-100, 100].

Figure 5.2.5 Residuals 4th Model

Figure 5.2.6 shows the residual comparison of all four models, it is clear that yellow area is smaller than others on vertical axis, thus the 4th model has the lowest mean residual.

Figure 5.2.6 Model Comparison

In figure 5.2.7, we compare residuals between our models and standard dead reckoning model. The dead reckoning results are collected from multiplayer mode with around 800ms latency and shown a mean residual of 372.07. We recorded ball vertical coordinate on one machine when the remote paddle moves and compare it with the result collected from the other machine if the same paddle moves. For better representation, we used absolute residuals in chart and sorted data in ascending order. From chart below, it is clear that dead reckoning has much higher residuals than behavioural prediction models.

Figure 5.2.7 Compare with Dead Reckoning

# 6. Conclusion and Future Works

There is no doubt that network latency will become lower and lower along with development on cable and router design, but at present, the best way to provide high game responsiveness is through player prediction. Maintaining consistency for these kind of methods are challenging, although no one has announced possessing such technology, but some researches like Microsoft Outatime have shown promising results.

Player profiling through regression analysis, at current stage, still cannot conclude its performance, but what we can conclude is that with enough data, it is quite feasible to predict player behaviours. Our results have shown that more data we take in, the more accurate the predictions are. Expect data volume factor, there are more rooms to improve our predictions by applying other approaches, such as using Weighted Least Square (WLS) instead of OLS we used in regression analysis. Currently, we are only able to predict when players move paddle in game prototype, to test the overall performance of our theory, we also need models to predict when players stop paddle, when players make mistake – move paddle to wrong direction, and how long players can fix their mistake. Even with all these models, we still need to consider that players are unlikely always stay in same level, they may play better as they keep playing, a self-learning mechanism is a very important factor that can improve prediction accuracy, all these we mentioned are needed to test our theory, but unfortunately we don’t have enough time to finish, that leaves some future works to us.

There are two defects in our theory and test we currently can think of. First, our predictions are built on individual profiles, which means all prediction models are player specific, due to the limitation on resources, we have only tested on one test subject, there is no way we can tell whether it works for other players or not before we have a trial result. Second, data collecting process may induce storage and bandwidth issues, which is something we don’t want to see.

Until now, we have only focused on accuracy of our models, but inconsistency is unavoidable in predictions. We also need to work out a way to fix inconsistencies without having impact on player experience in future stage.

# 7. Reference

Aggarwal, S, Banavar, H & Khandeleal A 2004, *‘Accuracy in Dead-Reckoning Based Distributed Multi-Player Games’*, Proceedings of the ACM SIGCOMM Workshop on Network and System Support for Games, NetGames'04, 161-165.

Armitage, G, Claypool, M & Branch, P 2006, *‘Networking and Online Games: Understanding and Engineering Multiplayer Internet Games’*, e-book, viewed 07 June 2015, http://flinders.eblib.com/patron/FullRecord.aspx?p=274342.

Bakkes, S, Spronck, P & Lankveld, G 2012, *‘Player behavioural modelling for video games’*, Entertainment Computing.

Becht, I 2013, *‘Improving meIRL-based prediction models in video games using general behaviour classification’*, Faculty of Science, University of Amsterdam.

Becht, I & Bakkes, S 2013, *‘meIRL-BC: Predicting Player Position in Video Games’*, Intelligent Systems Laboratory Amsterdam, University of Amsterdam.

Brun, J, Safaei, F & Boustead P 2006, ‘*Fairness and playability in online multiplayer games*’, Research Online, University of Wollongong.

Fiedler, G 2013, *‘UDP vs. TCP’*, GAFFER ON GAMES, viewed 17 May 2015, http://gafferongames.com/networking-for-game-programmers/udp-vs-tcp/.

Fiedler, G 2013, *‘What every programmer needs to know about game networking’*, GAFFER ON GAMES, viewed 17 May 2015, http://gafferongames.com/networking-for-game-programmers/udp-vs-tcp/.

Finn, A 2015, *‘Slow Network: Bandwidth VS Latency’*, Aidan Finn – IT Pro, viewed 26 Mar 2015, http://www.aidanfinn.com/?p=9566.

Gambetta, G N.D., *‘Fast-Paced Multiplayer (Part II): Client-Side Prediction and Server Reconciliation’*, Improbable, London.

Hargreaves, S N.D., ‘*Making Networked Games with the XNA Framework*’, XNA Community Game Platform Team, Microsoft.

Harrison, B & Roberts D 2011, *‘Using Sequential Observations to Model and Predict Player Behavior’*, North Carolina State University.

Herik, H.J.V, Donkers, H.H.L.M & Spronck, P.H.M 2005, *‘Opponent modelling and commercial games’*, G. Kendall, S. Lucas (Eds.), Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG’05), IEEE Press, Piscataway, New York, USA, 2005, pp. 15–25.

Keilson, J., & SpringerLink 1979.**‘***Markov Chain Models — Rarity and Exponentiality’***,** Applied Mathematical Sciences, 28.

Lam, S 2014, *‘Sliding Window Protocol and TCP Congestion Control’*, Department of Computer Science, The University of Texas, Austin.

Lee, K, Chu, D, Cuervo, E, Kopf, J, Grizan, S, Wolman, A & Flinn J 2014, *‘Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Cloud Gaming’*, University of Michigan, Microsoft Research & Siberian Federal University.

Liang, B, Chen, L & Cai H 2007, *‘Implementation of Dead Reckoning technique in network game’*, School of Computer Science and Engineering, University of Electronic Science and Technology of China.

Liang, D, Boustead, P, Cheok, AD, & Ishibashi, Y 2006. *‘Using local lag and timewarp to improve performance for real life multi-player online games’*, Network and System Support for Games Proceedings of 5th ACM SIGCOMM Workshop*,*37-Es.

Moon, K 2007. *‘Consistency Maintenance Algorithms for Multiplayer Online Digital Games’*, School of Information and Communication Technology, Griffith University.

Palmer, P & O’Connell D 2009, *‘Regression Analysis for Prediction: Understanding the Process’*, Cardiopulm Phys Ther J. 2009 sep; 20(3): 23-26.

Shanghai Airport Authority 2015, *‘Maglev’*, viewed 7 April 2015, http://en.shairport.com/2012-11/09/content\_15904433.htm.

Smed, J & Hakonen, H 2006, *‘Algorithms and Networking for Computer Games’*, John Wiley & Sons, New York.

Sykes, A N.D., *‘An Introduction to Regression Analysis’*, The Law School, University of Chicago.

VALVE 2014, *‘Source Multiplayer Networking’*, Valve Developer Community.

Webersdorfer, M 2012, *‘Player prediction techniques for AI in video games’*, FH Salzburg.

Wilcox, WR 2010*, ‘Explanation of results returned by the Regression tool in Excel’s Data Analysis’*, Clarkson University.

Wright, JR 2000, ‘*Beyond Equilibrium: Predicting Human Behaviour in Normal Form Games*’, Simon Fraser University.

Wu, N., & SpringerLink 1997.**‘***The Maximum Entropy Method’***,** Springer Series in Information Sciences, 32.

# Appendix A: List of Future Works

1. Finish models to predict when players stop paddle, when players make mistake and how long to fix mistakes.

2. Explore other approaches in Regression Analysis except Ordinary Least Square (OLS), eg. Weighted Least Square (WLS)

3. Test other independent variables in Regression Analysis except ball vertical coordinate, eg. Distance (ball to paddle)

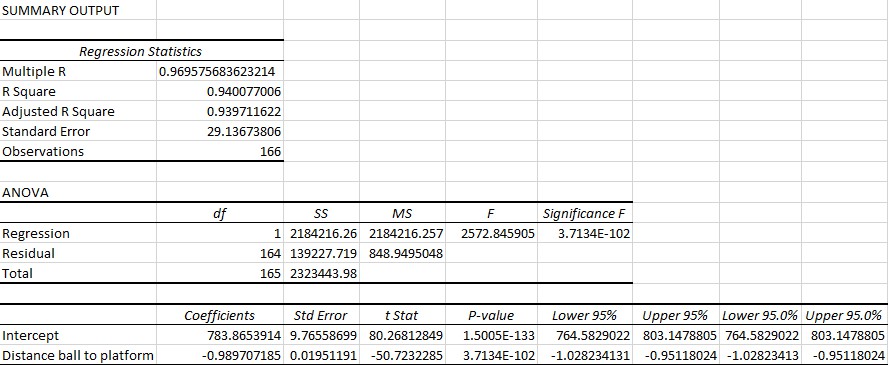
4. Make models can automatically analyse new data and update model after each play, test bandwidth issues might be caused by self-learning feature.

5. Set up trials and test on multiple subjects.

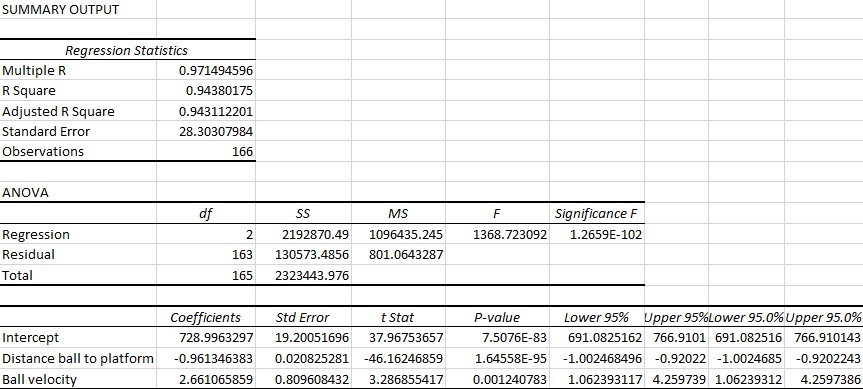
6. Test for inconsistencies and develop possible consistency models

7. Research on other factors that have influence on player performance except we already used in models.

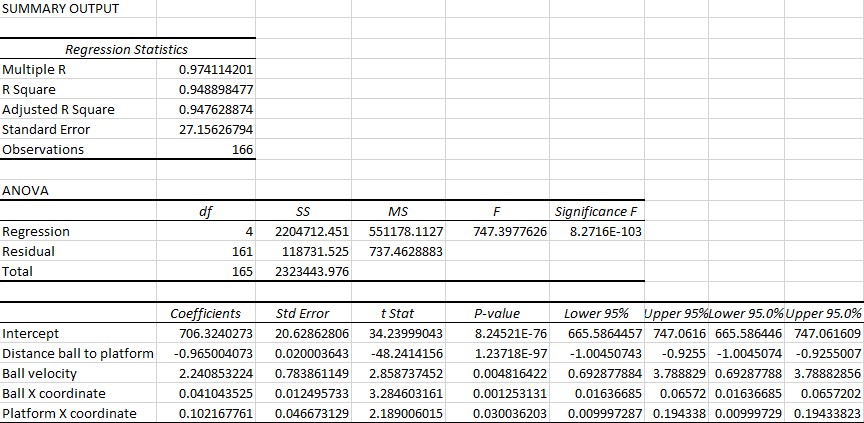
# Appendix B: Regression Analysis of Each Model from Excel

1st model

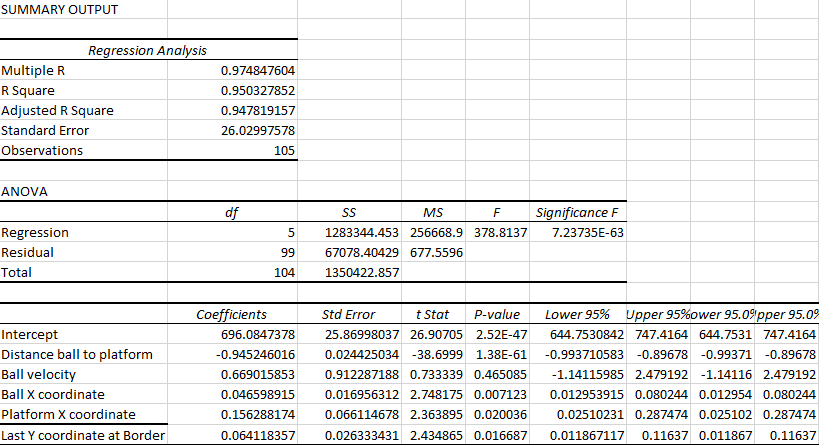
2nd model



3rd model



Discarded 4th model



4th model

