Performance Analysis In Network Games

Qin Lu

December 10, 2006
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Greger Wikstrand
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

Delay is an unavoidable problem in networks, and it always creates irritations for the network game players. An experiment was done with developing a simple Pong game to see how delay affects the game players. Moreover, a method named "decorators" was introduced into this experiment and it provides the feedback of the delay, and the effects of the feedback are also tested in this experiment. Through the result we find out on the low delay condition, players took more interest in the game with a decorator than without it, they adapt a certain amount of delay. But on the high delay condition, the feedback of the decorator did not have a good effect on the game. There should be a better method to get over it, and that is the study area of my future work.

Contents

List of Figures

Chapter 1

Introduction

Nowadays, for commercial reason, network games become more and more popular. Compared to common PC games, one of the advantages of network games is that the players do not have to play the game alone, they are cooperating with the whole world. Nevertheless, new problems are born in the games of which the most obvious one is delay. Delay leads to latency at the user layer in the network, that means it takes a while until information (e.g. about the movements of the opponent objects and their new positions) reaches the receivers. It leads to several complicated situations, there is an example in [1] where player A shoots player B in a shooter game and sees player B "die". On the other hand, player B has moved away and sees the shot misses, then what is the true version?

Obviously the most direct method for minimizing the problem is to provide a better quality network environment of service (see chapter 2) which is from the network perspective, but usually it is out of the game producers' or game players' control. From the application perspective the solutions focus on hiding the problems and from the usage perspective the solutions focus on making the problems more obvious for the users to adapt. Recently, researchers have focused on methods which improve the collaboration quality in shared user spaces, one of the methods is called "decorators" [9]. That is also the key word in this thesis.

In my experiment we looked at the effects of delay in a simple network game: Pong. Furthermore, we integrated the method "decorator" into the game to give the players a feedback to predict the future state of the object with a shadow. We grabbed some persons in the lab and asked them to play the game in different conditions. Then we made statistics with the data coming out from the game playing. We analyzed the result and drew some conclusions. We could see how the decorator affects the problems presented to the players.

1.1 Organization of this thesis

The thesis is organized as follows.

-**Chapter 1**: Introduce the problems and motivate the research effort.

-**Chapter 2**: Introduce multi-player games and the nature of delay, survey former works about the delay problems.

-**Chapter 3**: Introduce the application building and the processing of the experiment.

-**Chapter 4**: Analyze the experiment results.

-**Chapter 5**: Discuss on the result compared to the former works. Draw the conclusions and make a plan of future works.

Chapter 2

Nature of delay and approaches of dealing with delay in network games

2.1 Multi-player games

A multi-player game is a kind of video/computer game which multiple people online can play the same game at the same time against or with each other. It is a little different from the traditional video/computer game. In video/computer games, players play the games with themselves or the artificial opponents created by the computers, so it is a single-player game; or sometimes it is allowed multiple players to play but on a single computer or a single screen with different zones of a keyboard or several joysticks. However, a network game is quite different. It allows thousands of players online to play at the same time only requiring to gather around a single game system. Sometimes the players fight with each other or sometimes they make a team to achieve a common goal by defeating the opponents also created by the computers.

Figure 2.1 shows us a very popular multi-player game - Counter Strike. That is a team-based first-person shooter game. As of May 2006, it was still the most widely played online first-person shooter game in the world. Steam statics showed that there were over 200,000 players for it in 2006.

The word "multi-player" usually means that there are many players (at least two) playing together by connecting via a network, usually by a LAN or Internet, so we also call a multi-player game as a network game. Network game players tend to feel more enjoyable when they play via a LAN because a LAN essentially eliminates problems which are very common in internet, such as lag which refers to delays experienced in computing communications. Sometimes people hate the anonymous players because someone do not like to play with the people they do not know, but the most serious problem should be caused by delay of which the condition of a LAN is normally much better than Internet.

Building a multi-player game is more difficulty than building a traditional PC game.

Figure 2.1: A scene in the famous Network game - Counter Strike

New technical problems come out, the network plays an important role in the game. For example, you play poker game with another three persons online, you have to follow one by one. Sometimes the delay problems come out when you are right there having followed and waiting for someone else, it turns out to be that the others are waiting for you. It might really annoy all of you.

We have mentioned that there are many reasons probably leading to such a bad network condition, latency is one of them. Players often refer to latency by the term ping which measures the round-trip network communications delay. Sometimes it is because of packet loss and choke which makes a player unable to register his/her actions at the server, then those actions will get lost. For example, in a first-person shooter game, the problem is always the bullets seem to hit the enemy but he/she is not hit. The shoot action is lost. The point is the player's connection with the network is not the only factor, the entire network to the server is also relevant, and it also depends on the server's speed. While latency is frequently complained about, lack of finesse and decent tactics are probably more dangerous than a slow connection in most games. However, the variations in latency can be another story, these make it difficult to play the games.

Obviously, network games attract more people to play, and they also meet more problems. What we should do is to think about how to minimize these problems.

2.2 Definition of delay

Delay leads to latency which means the time between the generation of an event of a client and the result updating of the game state at the server. It leads to significant problems in network multi-player games.

How the delay problems happen results from two main sources: messages transmitting and the processing of these messages at the endpoints. The first one arises from a combination of transmission delay, switching delay, queuing delay and retransmission delay, while the second one arises from the sender, the receiver and the server.

Trans-Atlantic Path		
NY	74	London
Wash	104	Frankfurt

Trans-Pacific Path		
LA	110	Tokyo
LA	177	Singapore
SF	150	Sydney
SF	134	Hong Kong

Figure 2.2: Different Latencies from main US cities to the world

Latency and jitter are two aspects of the delay. Latency causes the communication a little bit slower with actions sometimes being after when they actually happen. Jitter is the variation in latency. It causes the results hard to predict.

Here are different latencies(ms) from some main cities of US to some other cities in the other countries across the Atlantic or Pacific ocean at 2:22, October 13th, 2006 (Swedish time) (source from http://ipnetwork.bgtmo.ip.att.net), (Figure 2.2). Because of the jitter, the latency was changing all the time.

2.3 Approaches of dealing with delay

One approach of dealing with delay is to reduce it, and another one is to make it more clear to make sure the users pay more attention on it. Here we introduce some of the solutions on dealing with it.

2.3.1 Network approaches

There are many approaches of reducing the delay on the network layer. One is buying better connection materials to enlarge the bandwidth, but the disadvantage is that it costs much money. There is yet another way to achieve the similar goal which is controlling the bandwidth via different network architectures.

Client-Server architecture

The most common architecture used in multi-player games is client-server architecture. It separates the clients from the server. The game simulation takes place at the server, rather at the clients. Each client sends the requirement to the server. The server receives the information and updates it, and then sends the resulting back to the client. Finally the client displays the updated game view to the player.

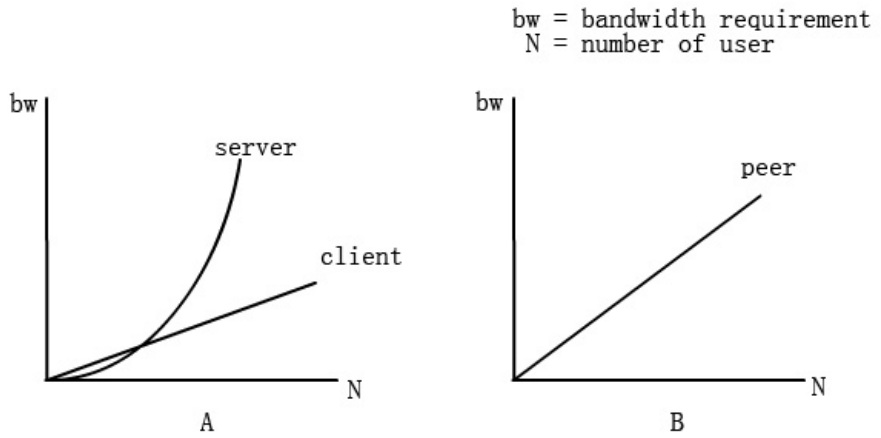

Figure 2.3: Bandwidth requirements in the CS and P2P architectures

We analyze the bandwidth requirement of the CS architecture. A client sends a requirement to the server in every player update period, so the client upstream bandwidth requirement is L_U/T_U. L_U is the size of each update (we assume all of the updates have the same size), and T_U is the update period of each player. The downstream bandwidth requirement is derived from the updates sent from the server. The server has to reply a message to each client, so the size of the global message L_G should be equal to NL_U. Then the downstream bandwidth is L_G/T_U. The total bandwidth is the sum of the upstream bandwidth and the downstream bandwidth, which is

$$\frac{L_U + L_G}{T_U} = \frac{(N+1)L_U}{T_U} \tag{2.1}$$

which scales linearly with N, the amount of the clients. The bandwidth of the server can be obtained similarly. The relationship between the server and the clients bandwidth requirements can be seen from Figure 2.3, A.

The bandwidth of the server is N times of the bandwidth of the clients. It scales quadratically with N, so the bandwidth requirement of the server becomes the bottleneck of the whole system.

Peer-to-Peer architecture

The network built with peer-to-peer (P2P) architecture is about sharing: giving to and obtaining from a peer community which gives some resources and obtains other resources in return [12]. It relies primarily on the computing power and the bandwidth of the participants in the network. It is usually used for connecting large amount of nodes for sharing content files, and so on. In this situation, each player is responsible for its own simulation. Since there is no particular server to detect and solve the inconsistencies problem, players have to detect and solve the problem by themselves with a distributed agreement protocol.

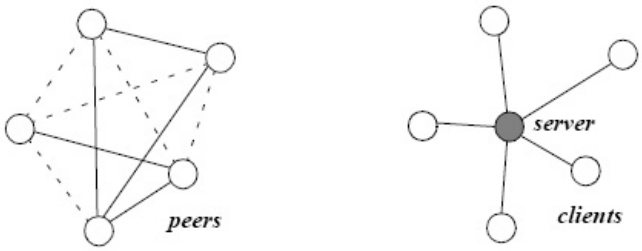

Figure 2.4: P2P architecture and client-server architecture

There are some advantages of the P2P architecture. It does not matter if some of the nodes have problems on transferring data, because in a pure P2P system, it enables peers to find out the data without relying on only one centralized server, but any available peers in the network. It means, there is no single point of failure in the system. Moreover, it reduces latency among peers [13]. But it is not as popular as a CS architecture in the game industry because it is difficult to implement for network games. Without a server in a multi-player game means the game company could not get the information about who are playing the game and how long they are playing. In that case, they receive the money from nowhere. But lots of files transferring or downloading software applications are built with the P2P architecture, like eMule, BitTorrent. That is because with this architecture, users can search data from any of the nodes in the network, they follow a global protocol in determining which peers are responsible for the data and then direct the search resulting towards the response peers, then they can obtain the data. Users could easily get more sources in the P2P architecture than in the CS architecture.

Again, we analyze the bandwidth requirement. In P2P architecture, each client sends and receives data from each other, so the upstream and downstream bandwidth requirements are the same. There are totally N-1 clients not including you, so both of the bandwidth requirements are $(N-1)L_U/T_U$. The sum of the bandwidth requirements is

$$\frac{2(N-1)L_U}{T_U} \approx \frac{2NL_U}{T_U} \tag{2.2}$$

which can be seen from Figure 2.3, B. The bandwidth requirement of a peer is about two times of the bandwidth requirement of a client in the CS architecture. But the advantage is that it removes the server bottleneck.

You can see Figure 2.4 (from Pellegrino and Dovrolis, "bandwidth requirement and state consistency in three multi-player game architectures") to know what exactly the P2P architecture and the CS architecture are.

PP-CA architecture

This architecture is designed by Pellegrino and Dovrolis [14]. They proposed a modified version of the P2P architecture which they called Peer-to-Peer with central arbiter architecture. It has the scalability properties of the P2P model and also owns the simple consistency resolution mechanism of the CS model.

In the PP-CA architecture, a client changes update data with each other just like in the P2P architecture. It minimizes the communication delay between clients. But a client does not send update data only to all of the other clients, but also the central arbiter. What a central arbiter does is to listen and detect if there are inconsistencies. If no, it keeps silence and does nothing, else, the central arbiter will resolve them and send the results back to all the players.

According to the same way, we obtain that the bandwidth requirement of a client is

$$\frac{NL_U + (N-1)L_U}{T_U} \approx \frac{2NL_U}{T_U} \tag{2.3}$$

which is still linear with the number of players. The bandwidth requirement of the central arbiter depends on the frequency of the inconsistencies happened. If it never happened, then the requirement will be NL_U/T_U; If a single state happened in each update period, then the requirement will be $2NL_U/T_U$ which is equal to the bandwidth requirement of a client in the CS architecture; The worst condition is that the inconsistencies occurred in each client and in every update period, then the bandwidth requirement will increase quadratically with N, the same situation as a server in the CS architecture.

2.3.2 Application approaches

Bucket algorithm

Bucket algorithm (input buffering) is a method for avoiding inconsistencies between players in a networked multi-player game. It is primarily intended for games without a central server using multi-cast to communicate between the players [5].

Bucket algorithm solves inconsistencies problem by letting the players experience the maximum delay in the game as long as the jitter is limited. Bucket algorithm divides the considering time into intervals of length T. When computing, it is a local view at the end of the internal i which has the length of $[i, i+T]$. An entity using all application data units is generated by the remote entities during the internal $[i-\Delta, i+T-\Delta]$ as well as its local state during the same interval to compute this local view. In that case, the actions occurring during the internal $[i-\Delta, i+T-\Delta]$ are grouped together in a same "bucket". The added delay Δ compensates the actual network delay, and after the units within Δ time have been sent, it will be processed in the "bucket". Though the game becomes consistent, the average delay enlarges.

MiMaze is a game which is unique in the area of multi-player games, being based on a server-less architecture together with distributed synchronization (input buffering) and dead reckoning based error control. In MiMaze, events are delayed for a certain

time that should be long enough to prevent disordering before being executed [15].

Dead reckoning

Dead reckoning method is based on a navigational technique of estimating one's position based on a known starting point and the velocity. The same idea can be applied to predict the data from the other nodes, which make it possible to prolong the interval of message transmissions and abolish the network latency at the cost of data consistency. When a message is received, the local data is updated accordingly. The message can also include some extra information (e.g. the velocity), which is used to predict the change of data over time. Alternatively. This information can be omitted and the known history can be used for extrapolating the data. Moreover, the transmission interval needs not to be consistent but the messages should be sent only when the dead reckoning exceeds some error thresholds [6]. Predictor displays work best when the behavior of the remote object could be predicted. The disadvantage is that the displays only show the predicted states, and players can not compare the state of now and future to adapt their behaviors.

In MiMaze, However, if events are lost or arrive later than expected, the game will not attempt to detect the inconsistencies. Dead reckoning is a prediction mechanism which predicts a next position based on the last position and movement patterns.

Physical limitations

Physical limitations means that the game designer make the inevitable delay part of the game by changing a little rules of the game.

For example, in a Pong game, the game designer can dig a "river" at the middle of the game field. It takes the ball a certain time to go across the river. It is equal to that the width of the river could be adjusted to compensate for the network delay. The disadvantage is doing this sometimes makes the game a little bit boring.

Delay feedback

This method will be discussed in section 2.4.

2.3.3 A three-Layer Model of Network Application

There are totally three layers in the analysis of a network application, which are network layer, application layer and user layer. Application layer is used for displaying information, having computational operations for the users, packing and unpacking data obtained from the network. Network layer is used for data transmitting from one node to another. User layer is where the actual users take place [3] (Figure 2.5).

Network approaches work at the network layer. Input buffering works at the application layer and the interface between the network and the application layers. Dead

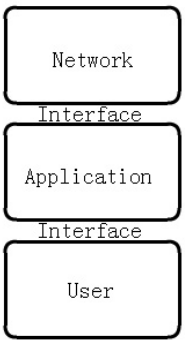

Figure 2.5: A three-Layer Model of Network Application

reckoning works at the application layer and the interface between the user and the application layers. Physical limitations work at the application layer but produce changes at the interface between the user and the application layers. Delay feedback works at the interface between the user and the application layers. Here we see that the approaches we proposed almost cover all of the layers and interfaces of the three-layer model except the user layer. But player is also an important character in the whole system, users can be part of the approaches by adapting their behavior with the "super computer" brain.

2.4 The feedback for revealing delay

Usually there are two ways to deal with the displays of the delay, one is to hide it, another one is to make users more aware of its presences and characteristics as to further encourage the adoption of coping strategies, "decorator" is designed for that. This is a virtual object added in the user's interface to enhance a user's understanding of an associated delay-induced phenomenon. It would also be possible to display quantitative or qualitative feedback of the delay during the game to allow the players to adapt appropriately.

There are many different kinds of decorators [7] to deal with the varied consequences delay for different applications, and there are two specific families of decorators: magnitude of delay decorators, past and future state continuum decorators.

2.4.1 Magnitude of delay decorators

The first family of decorators shows us the presence or magnitude of the delay which are also the basic information of the delay.

Round-trip abstract decorator. It is the simplest decorator which shows the overall round trip time for communicating with a remote object in which the total delay involved sending a message and receiving an immediate response.

Jitter decorator. It shows the variation in delay over time rather than the raw magnitude of delay. Usually it can be an important factor in predicting the movements

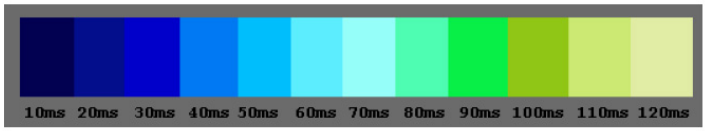

Figure 2.6: Color decorator

of the remote objects [8]. There is an experiment to represent the processing with a jitter decorator. The color starts to change when the client received a message, and every time an update message arrives, the color is reset and the timer is restarted [2]. Figure 2.6 shows that the color of the remote object changes as the jitter changes (from Shervin Shirmohammadi and Nancy Ho Woo, "Shared Object Manipulation with Decorators in Virtual Environments").

Temporal distance decorator. The decorator changes its state to reveal the magnitude of the delay in one direction, from the local user to a remote object or vice verse, because it is important to show the delay associated with a given direction dues to different down-link and up-link bandwidths or routes. So normally it is necessary to introduce two decorators for both directions.

Third-party delay decorator. The decorator provides an inter-subjective view of the delay between two objects which are both remote from the local observer.

2.4.2 Past and future state continuum decorators

The second family of decorators shows the past or future state of a remote objet in order to help the users to predict how to interact with them.

Obviously the past state decorator shows how the object appears in the past. The same, the decorator of the future state of the objects is derived of their past states, constrain of movements and current delays. The decorator helps users to predict the potential states of events so that they can plan their activities in advance. As the decorator is guessed as the most likely states of the future, we always use translucent or outline to represent the state of future in order to make the users understand the differences between the current state and the future state. It is a little similar as the dead reckoning approach. The dead reckoning only gives out the predicted displays, but with the future state decorator approach, the future state and the current state exists at the same time in order to compare with each other, giving the users more awareness of the delay problems.

In my experiment I used a shadow to decorate the future state of the paddle, it is clear to show the player where is the paddle supposed to be currently (Figure 2.7).

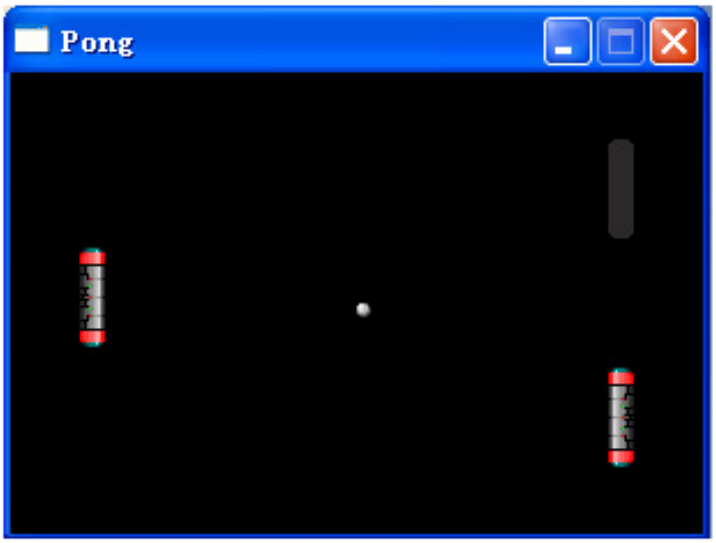

Figure 2.7: The decorator of shadow in the Pong game

Chapter 3

Experiment

3.1 Expectations

We expect that people take more interest in the low delay condition and it is better with the feedback; We expect the degradation of performance in the significant delay condition; We expect that players take less actions in the high delay and without feedback condition as they are trying to adapt the behavior in bad environment.

3.2 The Game

3.2.1 Game choosing

I used the Pong game for my experiment. There were three reasons: First, it could be easily emulated in the network environment, the latency could be expressed clearly on the paddle; Second, it was not too complicated to build, so it would not waste much time on building the game, anyhow, the procedure of the experiment was the most important part; Third, this game was used by Greger before, it is comparable using the same game to do a similar experiment.

3.2.2 Environment

The application is running on Microsoft Visual Studio .NET 2003, written in C♯.

In the programming reference, I used SDL which provides high-level accesses to game making in variety aspects. In the official web (http://cs-sdl.sourceforge.net/index.php/Pong) it provides an article to guide us through the creation of a simple pong clone. As a matter of fact, it became the frame of my Pong game. As it says, the article is so simple that it has only a ball bouncing in the screen even without paddles. What I do is to make it more interesting for playing according to the purpose of my project.

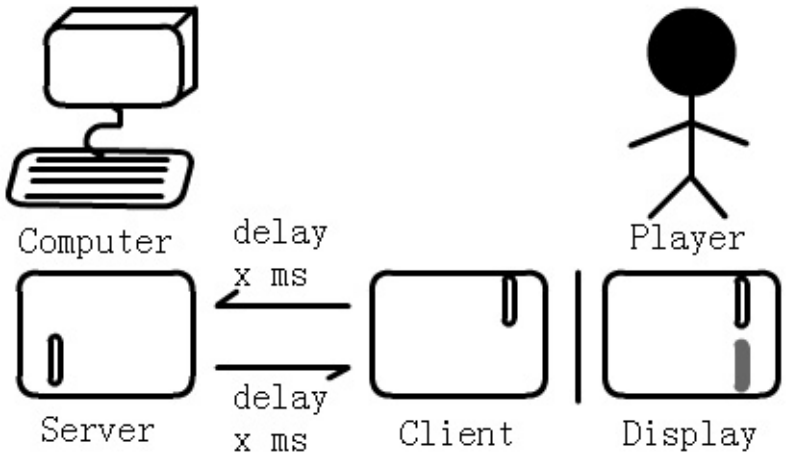

Figure 3.1: The game building structure

3.2.3 The game building structure

The experiment was setup with two separate computers. One of it was running a server application while another one was running a client application. It is the simplest CS architecture network building. We have discussed in chapter 2 that in the CS architecture the bandwidth requirement increases faster than in the P2P architecture but in the P2P architecture it reduces the server bottleneck. Yet the CS architecture has another advantage is that it is easier to maintain the consistency and reduce the treating. If a client is required to report its location and game using condition, it is easy for it to fake the real information, while obviously a central server can solve this problem better. I think this is why most games are built in the CS architecture other than in the P2P architecture, so we also use the the CS architecture in our application.

In the game, one of the players is controlled by the computer, so in fact the score of computer is exactly the amount of balls you have missed, it can be used to evaluate the level you master the game.

The experimental setup can be seen from Figure 3.1.

3.2.4 How the application works

One of the aims of my project is to decorate the delay in the network game. In that case, what I do is to make both a server and a client connected followed the TCP protocol.

In the game, we make some actions by moving or clicking the mouse, the actions are sent to the server in data packets. In the processing of sending, we control and emulate that there is a certain delay on the way. Then the server updates the data, makes its

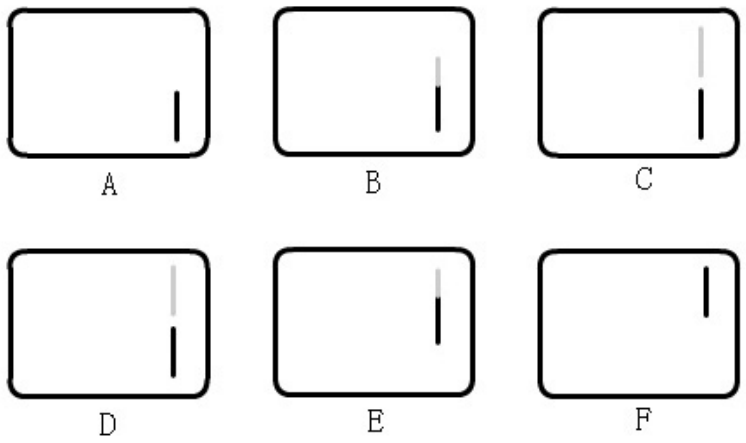

Figure 3.2: The shadow decorator state

legal response and then sends it back to the client, of course there is still a "delay" on the return way. After that, the result can be seen from the display. We suppose that the delay on the sending way and the receiving way are the same, they are both x ms, so the total latency should be $2x$ ms, the actually latency could be ignored because the experiment was taken in a lab, the latency crossing the local LAN is very tiny.

That is how the game works. We used two threads for handling the data. From the client's view, one for sending data, and one for receiving data, so the data will not block to each other.

The game's field is a 300*200 window with a ping ball and two paddles on both left and right sides. The paddles are able to move up and down but not forward or backward. The ball initially stays in the middle of the playing field, the game starts when clicking left key of the mouse. When the game starts, the ball starts bouncing around hitting the paddles or the walls. Each time you miss hitting the ball back, the ball hits the vertical wall on your side of the playing field, then the other player scores a point, but the ball will continue bouncing around and never stop until the game is going to be over after 30 seconds. Considering it as a game simulating table tennis in reality, I set the random rebound angle because you will never know how your opponent hits the ball, unlike some implementations where the angle depends on the position the ball hits on the paddle.

As a matter of fact, the future state decorator is the soul of the experiment, the shadow shows the predicted state of the paddle, it is not actually existing, the players can not hit the ball back with the "shadow", it is only a decorator to remind the players that the paddle will move to this position in the future with in a short time, the time depends on the latency, see the Figure 3.2.

From Figure A to figure F are the game states from time t_0 to t_5, it starts with

t_0. At that time, the decorator is hiding beneath the paddle. at t_1, the player moves the paddle up. Because of the existing latency, the paddle is not moving immediately, but the shadow is. It moves to the position where the paddle is supposed to be. At t_2, the player keeps doing some actions, and the shadow keeps moving to the top of the game field. At t_3, the player stops controlling, the shadow also stops, but the paddle is still not moving. At t_4, the paddle moves, so the totally latency should be t_4-t_1. The paddle keeps moving until goes to the same place where the shadow is and covers it.

The game can be downloaded from my home page: www.cs.umu.se/~int05.

3.3 Procedure

The experiment was divided into three parts. The first part was taken before the formal experiment started, the participants were asked to answer some questions about their age, education, gender, etc. In the end of the experiment, we could know the general conditions and backgrounds of the people who took part in this experiment. The second part was playing the game, that was also the main part of the experiment. The third part was taken after playing, the participants were asked questions about their enjoyment, etc (see "Dependent Variables" below).

Each player was asked to play 6 rounds of the game. The first 2 rounds were used to make the players get used to the game in order to make the results more sensible. To achieve a counter-balanced design, every participant was asked to play the game with different conditions in different orders.

3.4 Independent Variables

There are totally three independent variables: delay, feedback and order.

Two levels of delay were introduced into this experiment: 50 ms and 250 ms. Since the delay was happened resulted from both input and output so the latency was approximately twice of the delay, it should be 100 ms and 500 ms.

There were both conditions of game with feedback and without it. The feedback is decorated as the estimating position of the player's paddle with a "shadow". There was no decorator for the ball and the paddle controlled by the computer.

In order to make a balance of the experiment result, there were different procedures for each player to minimal the order effect. The method is to make 16 participants into 4 groups, each group has its own order for playing the game. You can read the table 3.1 (for example, the 1st person was made into group A and the 2nd was made into group B, like so on) about how to make up the groups. You can read the table 3.2 about what is the mission in each group (N = no feedback, S = with shadow feedback, 100 or 500 means the amount of latency).

Table 3.1: Group making

Group				
A	1	8	11	14
B	2	5	12	15
C	3	6	9	16
D	4	7	10	13

Table 3.2: Experiment order

Group	1	2	3	4
A	N100	N500	S100	S500
B	N500	S100	S500	N100
C	S100	S500	N100	N500
D	S500	N100	N500	S100

3.5 Dependent Variables

3.5.1 Enjoyment

Enjoyment is used for measuring of fun and enjoyment of the players by asking them how much they enjoyed the game. It is also an important value for making the games or entertainment applications. There are many methods to measure the value of fun, including: observation, self-report and physiological measures.
In my experiment, we used self-report measure method. The participants were asked about how much they enjoyed the game around a seven-point scale, they were "very boring" , "boring", "a little boring", "not boring but no fun", "a little fun", "fun", "much fun".

3.5.2 Game score

Game score is used for evaluating how the players master the game. In my Pong game, when a player misses a ball, he/she gets a point, then the times of ball missing totally in a round (in 30 seconds) is the final game score of the player. It is said that the higher score you get, the worse you master the game.

3.5.3 Paddle movements

Paddle movements is used for measuring how long distance the paddle totally has moved in the game. The time of each round in the game is the same, which is half minute, so it means the longer distance of the movement is, the more often the actions happen.

3.6 Participants

Participants were grabbed in the campus of Umeå University. Each of them was asked to fill in an information table. The content included their name, gender, age, education,

experience of playing game, etc. In the end of the experiment we knew that there were totally 18 persons doing this experiment, for some reasons (like someone who did not play the game carefully), the data of two persons among them were unavailable, so I did not use their data and kept grabbing persons to do the experiment until I obtained 16 of available data. The participants were average 26.3 years old, 10 of them were female and 6 were male. Most of them were university students and had experiences on playing computer games. That means they had enough ability to adapt the computer behavior in a short time, and did not need to have much training.

Chapter 4

Results

4.1 Enjoyment

From Figure 4.1 we know that there are significant effects of both delay and type of feedback to the enjoyment. Obviously the participants took more interest in the low delay (4.475) condition and with the shadow decorator (4.3) than in the high delay (3.725) condition and without feedback (3.781).

4.2 Game score

From Figure 4.2 we know that in low delay condition, the average score decreases from 1.0 (without feedback) to 0.78125 (with feedback), decreased by (21.75%). While in high delay condition, it plays an opposite effort, from 3.9375 (without feedback) to 4.60375 (with feedback), increased by (16.92%). But it is clear that the average score in high delay is always higher than in low delay. Pay attention, the higher the score is means the worse the player plays.

4.3 Paddle movements

There are significant effects of delay and the type of feedback on paddle movements. Paddle movements decrease with delay increasing from 414.51562 to 334.03125. They are also lower with shadow (359.85937) than without it (388.6875), (in the lower delay condition, it decreases from 432.65625 to 396.373). If the paddle never stopped in the game, the maximum paddle movements are about 1000. see Figure 4.3.

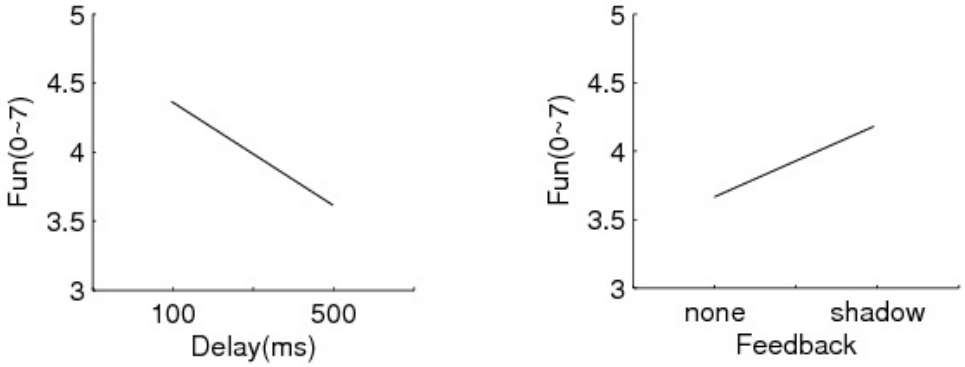

Figure 4.1: Main effects on self-report 'fun'

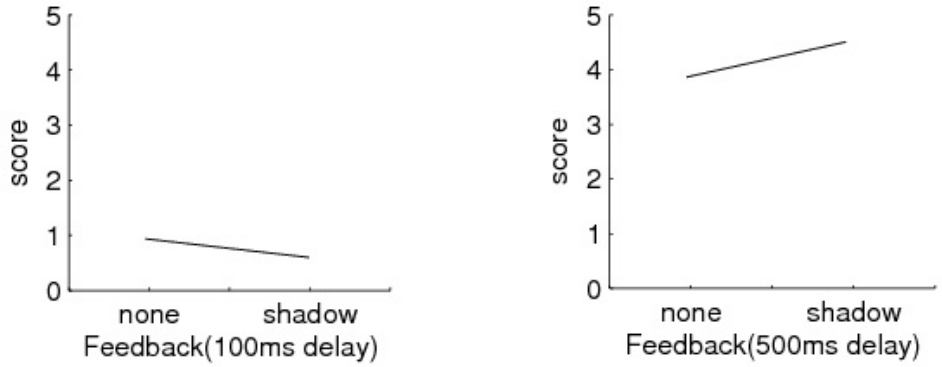

Figure 4.2: Main effects on game score

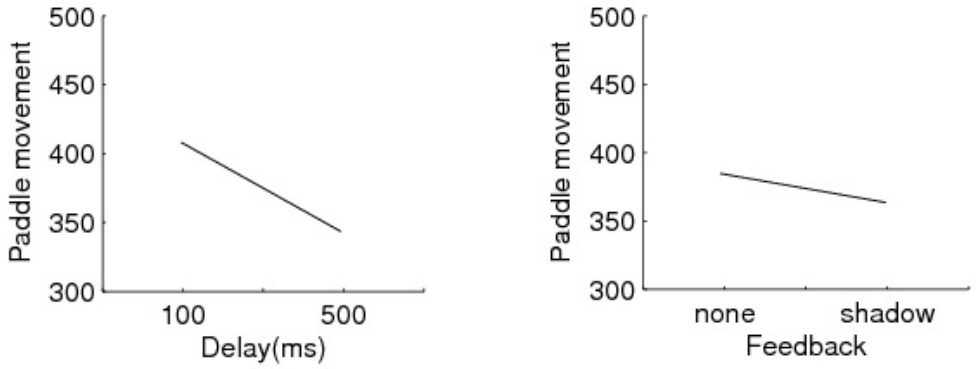

Figure 4.3: Main effects on paddle movements

Chapter 5

Discussion and conclusion

5.1 The former work

Greger Wikstrand, Lennart Schedin and Fredrik Elg [9] gave three hypotheses before they did their Pong game experiment in a simulated mobile phone: "Delay effort", "Delay action" and "Delay performance". The experiment put eyes on significant effects on four independent variables: enjoyment, mental effort, net distance and paddle movements. Analyzing the result they gave the conclusions, each of their hypotheses was finally identified.

Still, Allen and Breckler [10] did an experiment on Pong game even earlier, they introduced the delay of (0, 66, 133, 200 ms) in the game, and found that the enjoyment increased from 0 ms to 133 ms and then decreased at 200 ms.

Vaghi, Greenhalgh and Benford [11] also performed a similar experiment in a virtual reality environment. They used even more different levels of delay (0, 50, 100, 125, 150, 200, 300, 500 and 900 ms), they found that delay under 150 ms did not affects the players. Delay between 150 and 300 ms made the players miss some balls. At 500 ms it made the players have serious problems. Delay at 900 ms, the game was nearly unplayable.

From those experiences I considered that with different levels of delay, the decorators might have different kinds of effects on players adapting the behavior. That is also what was not mentioned in Greger, Lennart and Fredrik's experiment. So the main point of my experiment was trying to find out the different effects of delay between low and high delays.

5.2 My work

The results show us that with the increasing delay, people felt less enjoyable on the game, it might because of the gradual loss of control. Some of them said that they did not like the game to be too difficult in the end of the experiment. But with the shadow decorator, participants took more interest in the game, the feedback made them feel eas-

ier to play the game. So people felt more enjoyable when there was a feedback than none.

Analyzing the game score we found out that in the low delay condition, the feedback had a positive impact on the game, while with the high delay, the feedback had a negative impact. We would have thought that in the low delay condition, the feedback made the players know clear about when the delay happened and how long the delay lasted. But the situation in high delay would be severe, then the feedback did an opposite effort on it. The feedback did not make the player know clear about the delay but confused. The delay was so strong that they had no time to take over the problems they met, the shadow decorator might not help them to show the predicted position of the paddle but confused them mixing with the original object, So at last the result showed us an opposite impact.

The participants were able to adapt the behavior by decreasing their paddle movements when the delay enlarged. If the decorator should have made players feel easier to play the game, then participants would able to adapt the behavior without feedback by decreasing paddle movements, but the result was not. We can imagine that the same problem also comes from the high delay condition. They did even less movements when there was a shadow feedback. So we were sure that with the high level of delay, decorator really made the players confused.

With low delay, the feedback really made an positive effort on the game, nor with high delay.

5.3 Conclusions

There are totally three expectations we made before the experiment, and through the experiment most of them had been identified. The results confirm that the players felt increasing enjoyable in the low delay condition and with feedback, they confirm the degradation of performance in conditions with significant delay. But there is a paradox between the result and the third expectation (more actions in the low delay condition and with feedback), there were more actions in the low delay condition but not with the feedback. From the result we prove that the decorator feedback only had the positive effect on the lower delay condition but not in the high delay condition. So that was why the paradox come out. We can use a figure to represent the players' performance in the different conditions, (Figure 5.1) (1 means the condition they performed best, and 4 means the condition they performed worst).

5.4 The future work

According to the conclusions, we have not solved the problems of how to make the players feel more enjoyable and adapt the behavior better in the high delay condition. So there are two more things I have to keep on my future work. One is what level of delay is the balance point. In my experiment I used only two levels of delay: 100 ms and 500 ms. The results are definitely different between them. I do not know if there is a linearly increasing of the problem, or if there is one or more balance points. The second one

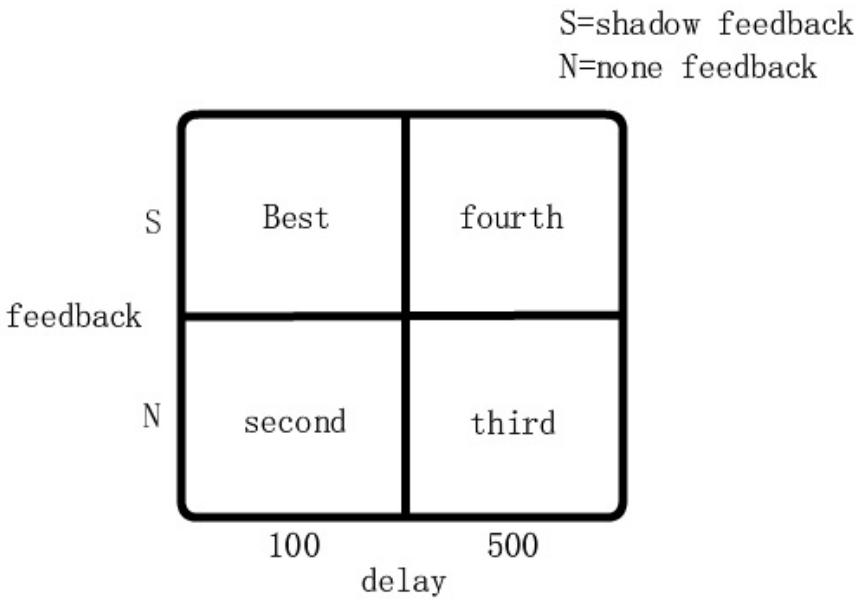

Figure 5.1: The order of players' performances

is how to solve the high level of delay problem with a decorator, is there a better decorator except using "shadow" which sometimes make the players not clear but confused?

Chapter 6

Acknowledgement

I would like to thank my supervisor Greger Wikstrand, he kept offer the biggest help to me in the whole period when I wrote my thesis.

I also would like to thank my parents, they always encouraged me though they were very far away from me.

I am grateful to Per Lindström, when I was a new student here and started my project, I knew nothing, he was the only man I would ask to for help.

Finally, I'm thankful to all the friends in Umeå and the participants of my experiment, I didn't even know some of them before. Actually they are all part of my thesis.

References

[1] Springer Berlin, Heidelberg, "Interactive Distributed Multimedia Systems and Telecomminicaiton Services: 7th Internationcal Workshop", IDMS 2000, Enschede, The Netherlands, 2000.

[2] Shervin Shirmohammadi and Nancy Ho Woo, "Shared Object Manipulation with Decorators in Virtual Environments", Distributed Simulations and Real-Time Applications, 2004, DS-RT 2004, Eighth IEEE.

[3] Greger Wikstrand, "Improving user comprehension and entertainment in wireless straming media, licenciate thesis Streaming video quality", page 3-11, Umeå University, Umeå, Sweden, 2003.

[4] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RFC1889: RTP: A Transport Protocol for REAL-Time Applicaiton", Internet Engineering Task Force, Internet Draft, 1996.

[5] L. Gautier, C. Diot, and J. Kurose. End-to-end transmission control mechanisims for multiparty interactive applications on the internet. In Proceedings of INFOCOM'99: Conference on Computer Communications, volume 3, pages 1470-1479, Piscataway, NJ, USA, 1999. IEEE.

[6] Jouni Smed, Timo Kaukoranta and Harri Hakonen. "Aspects of networking in multiplayer computer games", The Electronic Library, Volume: 20 Issue: 2 Page: 87-97, Apr 2002.

[7] C. Gutwin, S. Benford, J. Dyck, M. Fraser, I. Vaghi, and C.Greenhalgh. "Revealing Delay in Collaborative Enviornments", Proc. ACM CHI 503-510, 2004.

[8] C. Gutwin, "Effect of Network Delay on Group Work in Shared Workspaces", Proc. ECSCW 2001, Bonn, 299-318, 2001.

[9] Greger Wikstrand, Lennart Schedin and Fredrik Elg, "High and Low Ping and the Game of Pong", Sweden.

[10] R. B. Allen and S. J. Breckler. "Human factors of telephone-mediated interactive electronic games." In Proceedings of the 1983 ACM SIGSMALL symposium on Personal and small computers, 200-205, 1983.

[11] I.Vaghi, C. Greenhalgh and S.Benford. "Coping with inconsistency due to network delays in collaborative virtual environments. In Proceedings of the ACM symposium on Virtual reality software and technology, 1999.

[12] D. S. ilojicic, V. Kalogeraki, R. Lukose, K.Nagaraja, J. Pruyne, B. Richard, S.Rollins and Z.Xu, "peer to peer computing", HP Laboratories Palo Alto, HPL-2002-57(R.1), 2003.

[13] C. Diot and L. Gautier, "A distributed architecture for multimplayer interactive applications on the Internet", IEEE, 13(4), Aug 1999.

[14] J.D.Pellegrino and C.Dovrolis, "bandwidth requirement and state consistency in three multiplayer game architectures", Network and system support for games, Proceedings of the 2nd workshop on Network and system supprot for games, 52-59.

[15] L. Gautier, C. Diot, "Design and Evaluation of MiMaze, a Multi-Player Game on the Internet," icmcs, p. 233, 1998 IEEE International Conference on Multimedia Computing and Systems (ICMCS'98), 1998.