JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 7, NO. 6, JUNE 2004(pp. 814-823)

A Real Time Multiplayer Network Game System Based on a History Re-Transmission Algorithm

Seong-hoo Kim†, Kyoo-seok Park††

ABSTRACT

Current video games and game room games are played as a single player mode on the basis of various emulators. With the evolution of data communications and game technology, a new trend in the game industry has made the primary interests of game developers and companies in the game industry be moved toward a multiplayer mode from the traditional single player mode. In this paper, we represent how to implement a network game platform by allowing network modules to be run in conjunction with the current video emulator games. It also suggests a synchronization scheme for real-time game playout and practical mechanism that can support network games to be played with the Peer-to-Peer process using a lobby system.

Keywords: Network Game, Synchronization, Multiplayer, Real-time Game, Multiplatform, Latency, History Packet, Game Server

1. INTRODUCTION

A lot of people have produced and enjoyed various types of games, even before computers were introduced. Since computers have been common, many games mainly in the form of single player mode have first drawn public attention in the market. Thereafter, the technical evolution in the field of communication and game changes traditional concepts of games so that many game developers and companies started to move their focuses from single player mode to multiplayer one.

In general, single player games can be classified into PC games and video games, and multiplayer games in return into on-line games and network games. The current trend is that network games have much more interest than ever and other types of games[1,2]. It can be indirectly verified from real tendency like that Microsoft at present sells the network-based X Alive. The Counter Striker is famous for adapting on-line concepts from a mere PC game, this kind of games are adapted to on-line game from only one kind of game[2].

Current video games as well as those serviced at commercial game rooms are executed on a PC with suitable emulators in single player modes. Game companies try to provide on-line services for their existent single player mode games. But, unfortunately, current various games are not successfully serviced in accordance with their intention. In this paper, we suggest how to graft a network module into existent emulator games that can support video games in order to make them be run as network games, and introduce the concept of a distributed lobby system to reduce loads to a game server and to allow players to enjoy games like Peer-to-Peer network games. It also proposes a synchronization method on how to execute games in real-time.

2. RELATED WORK

This section presents various network topologies

※ Corresponding Author : Seong-hoo Kim, Address : (631-701) 449 Wolyong-Dong, Masan, Kyungnam, Korea. TEL : +82-55-249-2650, FAX : +82-55-249-2650, E-mail : arrayiv@csc.ac.kr

Receipt date : May 7, 2004, Approval date : June 29, 2004

† PhD course graduate student at Kyungnam Univ., Dept. of Computer Engineering.

†† Professor, Dept. of Computer Engineering, Kyungnam Univ. Masan, Korea.
(E-mail : kspark@kyungnam.ac.kr)

※ This research has been funded by the Kyungnam University, Masan, Korea.

for games. Also, categories used to classify games are briefly surveyed.

2.1 Network Topology for Multiplayer Games

The following Fig. 1 shows the network topologies for multiplayer games. Each of the 4 types affects game play with its generic characteristics[3].

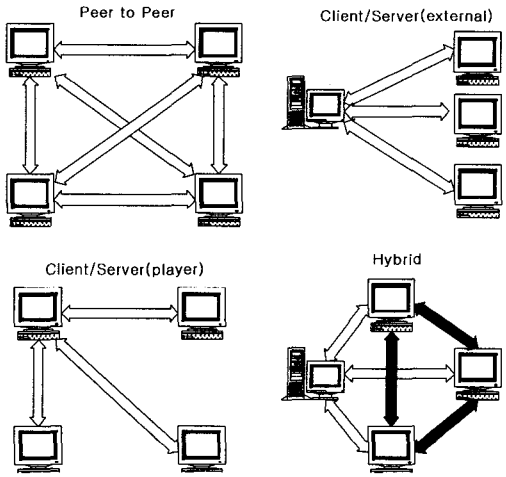


Fig. 1. Topology of Multiplayer network game.

■ In the Peer-to-Peer configuration, each player communicates with other players of the game. Typically, an external server takes a role in providing new game contents. Peer-to-Peer games have the advantages of that there is no need to maintain a game server, nor to design and code anything for the server. Each peer can communicate with each other without any arbitration by any server. This model can reduce the lack to a minimum. It is pointed out as the demerit of this model that the number of players is restricted due to the available bandwidth[2].

■ In the Client/Server(Player) configuration, one of game player joins in the game as a server. This player, not a dedicated game server, does not actively take part in the game. Like the Peer-to-Peer type, any external server is not required, but in return bandwidth-related problem is not considerable for this model because each player communicates with others who are connected to the server as role players. In case the server is disconnected or game is over, a new server is needed to substitute the old server. When selecting a new server, it should be considered that the best candidate is who can minimize the lack and bandwidth[2,3].

■ In the Client/Server(External) configuration, typically a dedicated server is located in a data center that is in general directly linked to a high bandwidth backbone line. Therefore, the server can provide high bandwidth services, but conversely the overhead cost of maintenance and dedicated line for this type is much higher than other configurations[3].

■ In the Hybrid configuration, 2 approaches are possible: load balancing scheme and web server scheme. In the load balancing, connection to servers is balanced among multiple servers with the control of a main server. With this approach, the main server runs a game, so that some limitations should be prepared when the number of game players increases even though connection is distributed to multiple servers[4].

2.2 Current Method of Playing Games

■ Non-network games are based on single player mode. This method imposes restrictions on scripting scenarios, but inherently no ones on network loads. It means that good scenario is the only requirement for appropriate game services.

■ Video games consist of a decoder to fetch CPU instructions, timing generator to control timing, display for sound and video, and I/O interfaces to input data from controllers such as a joystick. Most video games are played based on these elements[5].

Current on-line video games have been developing in the form of single player mode, and requires somewhat a good deal of cost because they are designed and developed by recomposing game ROM Images to fit the images into a system playing the game. They also have the drawbacks of, such as, when a new game is added, another single platform should be reconfigured. The suggested method in this paper allows ROM Images to be used even without redesigning them, and presents synchronization mechanism on how to adjust packet re-transmission following any error and/or loss by fluctuating network loads. And, multi-platform based channel server is also introduced in this paper.

3. SUGGESTED STRUCTURE OF NETWORK GAMES

This section constructs a server that can support both PC game and network game modules and game clients by games. Current single player game should have add function to install required network modules and practical synchronization mechanism, and secure consistency of game play after network modules to support real-time network play are added.

Video games mainly include console games and game room games, and various genres of the games are serviceable because they are implemented by game emulators. When considering the above situation, lobby systems are required for certain selected games to support various platforms of the games and allow the games to be played on network.

In order to support current games to be played as multi-platform network games, a network engine and server engine need to be implemented to have general purposes for network games, by, for instance, SDK. Also, a proper network game model that can minimize network delay and best handle packet loss on network is required.

3.1 Selection of a Current Game for Playing It on Network

Current PC games are implemented with various formats and forms so that it is hard to say that they are played in the same manner. Furthermore, real synchronization of the game play is almost impossible if open sources are not available. These are barriers preventing current PC games from seamless playing on network.

Regarding video games, the method in which demands for instructions and data are delivered from a CRT chip to CPU every 1/60 second is a clue to reprocess game play when interrupt mechanism is also applied. Open sources of emulators supporting video games make network engines to be installed to a game server. This type of games directly supports synchronization mechanism so that real-time game play is possible. An adaptive synchronization scheme is also available by game genres to handle variant network delay.

3.2 Game Architecture for Distributed Network

The game architecture in this paper adopts a hybrid model with which users can directly connect with a game server, and supports load balancing. Once a game starts, clients can with one another play the game regardless of any load on the game server because games are played on the ground of Peer-to-Peer concepts.

The game server consists of a multi-platform game middleware and game channels. Game clients include a game host to proceed with a game and a number of clients.

The suggested game structure is presented in Fig. 2.

The Middleware arbitrates a game server, and connects and disconnects a session established with clients by a TCP connection. It also supports information exchange between a server and client. The Game Server is a type of supporting variant emulator platforms.

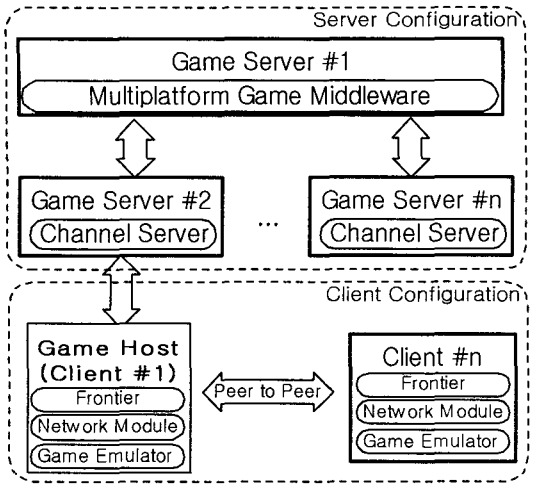


Fig. 2. Structure of distributed network game system.

Each Client can create a game room in the game server, and at the same time the client who created the room is designated as a host, which can control and manage the room.

3.3 Middleware of Network Game Server

The multi-platform game middleware is positioned between game players and a game server that arbitrates a game. A game player selects a game platform, which is the existent single play mode game supported by the game server, then goes through a login process. Once a game play passes the user authorization, he/she can register his/her own game resources(image list of game ROM) into the server. The resources are the images of the game ROM to be shared among players. An authorized player can check and verify the load of the game server that is registered into the multi-platform middleware, then transfer a game server list to players. Players can connect with one of the game servers.

The resource manager manages the image list of the game ROM. When required game ROM by a certain player is not available, the manager allows the resources from all of the users, other than the game server and players who is currently in playing the game, to be shared with the Peer-to-Peer process.

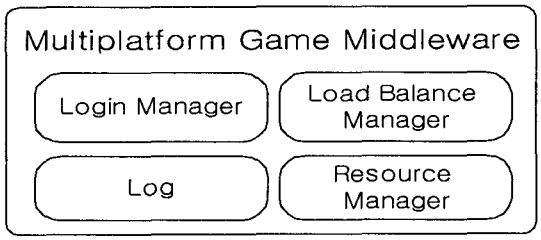


Fig. 3. Structure of the Multi-platform Game Middleware.

3.4 Network Game Server

The Network Game Server consists of the Lobby Manager, DB Server, Game Channel Manager and Chat Manager. The Fig. 4 shows a network game flow. The network game server helps a player to find out counterparts on network, controls chatting channels, has the invitation function for chatting and creates and manages game rooms. Any game player has to join a game room to play and enjoy a game. And, if the player does not have the images of game ROM, he or she can download them from the server or other players with the Peer-to-Peer mechanism.

After joining in or creating a game room, a player can exchange messages between or among players without any intervention by the server. The role of the Lobby is needed to support finding out game counterparts and joining in a game room. When required, a player can meet other players at the lobby in a designated computer.

- Game Channel Manager
  - Has resource control capability through which any player can download ROM images with the Peer-to-Peer process, and wholly manages player's resources.
  - Consists of channels according to the images of game ROM
  - Connects a selected channel to the lobby manager

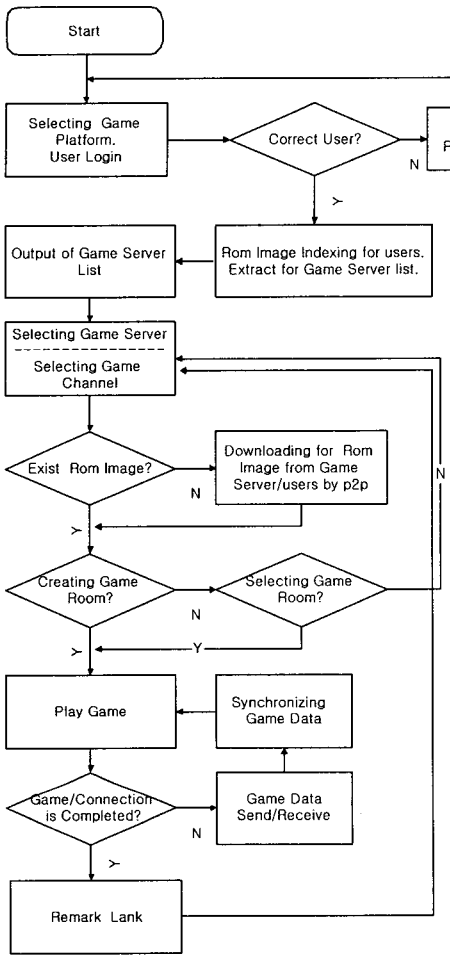


Fig. 4. The Flow of Network Game.

- Lobby Manager
  - Controls the game server and monitors the load of the server
  - Identifies the IP address and ID of a game player
  - Controls session with clients
  - A game host first opens one session
  - The lobby server is that plays a role of lobby. The lobby server only takes a role of arbitration.

- Chat Manager
  - Controls communication among game players.

With the Peer-to-Peer architecture, one player takes the role of a host and all other players play as clients. A client sends out all of input information to other clients as well as the server. The result of a game is evaluated by clients.

Consequently, network and message traffic increases proportional to the number of clients and so, clients are restricted to a certain degree. A host player controls basic management, such that checks how many players are connected with network. When a host player leaves a game, one of other players can take over the role.

3.5 Network Game Clients

The following Fig. 5 shows a architecture of game client, and it consist of the Frontier, Session Manager, Network Module and Video Game Engine.

The Network Module processes user input, and creates and analyzes packets based on threads. It also keeps a Synchronization Manager, which stores user input and output into I/O Buffers. The manager decides packet re-transmission and latency time after measuring packet losses.

The Synchronization Manager calls sequential information stored in I/O Buffers through invoking an interrupt using the emulator's Callback functions in order to synchronize input contents with the game engine.

The Communication Manager can create or delete any communication channel between game

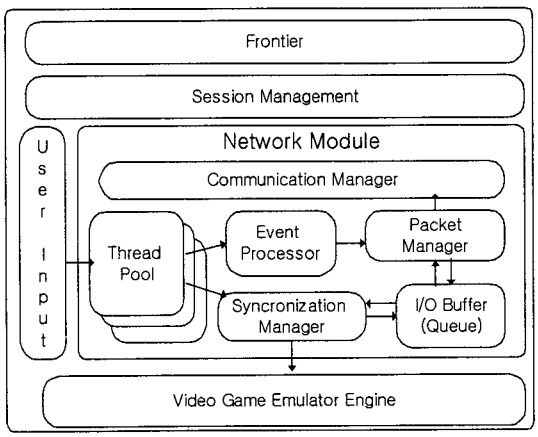


Fig. 5. The Architecture of a Game Client.

players supporting the Peer-to-Peer or Peer-to-Peer(Server) architecture, according to its system structure.

The Game Session Manager controls information on game players. When a connection is established for the first time, it detects a latency time to apply the time to synchronization for a network game.

4. TRANSMISSION CONTROL ON NETWORK

4.1 Network Transmission Model

When real-time games such as action games or shooting actions are to be played with TCP connections, blocking due to increased network load and latency can prevent games from seamless playing. Under a UDP transmission, packets can be lost and data distortion may occur. In order to prevent these problems, this paper suggests a history transmission mechanism that can support re-transmission of game data and CRC32 check to minimize re-transmission of game data due to packet losses.

The suggested system can send out data directly to a certain client, not necessarily through the centralized server. This mechanism is of help to reduce the loads on the server so that the stability of the server and fast data transmission can be secured.

4.2 Synchronization of Network Game

When implementing the synchronization of a network game, data re-transmission due to packet loss, network latency and presentation time, etc. should be considered as follows.

- Latency means the time for a certain computer to acquire required data packets from another computer through a computer network, and also considered as a function of paths occupied by data packets transferred between computers. When latency time measured at the network game server side is greater than half of the packet transmission time from a client, the time for a message from clients that arrived last at the game host as replies to a sync signal from the game host is decided as a latency time.
- Regarding emulators for video games, interrupt mechanism is used to control timing sequence adequately. A scan line interrupt to a CRT chip needed for timing control is a demand for data or instruction to CPU from a CRT chip every 1/60 second. When this interrupt is properly used to control timing sequence, network latency can be handled effectively. When a vertical retrace interrupt is invoked, a CRT controller is fully synchronized so that the emulator also has the same Callback functions as those of a CRT interrupt. The Callback functions are implemented with separate subroutines, performs input and output synchronization and controls a frame rate. If these scan lines are used to control timing, the playout by network latency can be adjusted.
- The contents of a message, which is exploited for a synchronization process, consist of input and output data. For any input data from keyboards or joysticks, 2-byte status information is delivered to the server. CRC32 checksum is also used to check packet loss during data transmission.
- If any packet loss or data distortion occurs, data re-transmission is inevitably followed to handle the situation. In the suggested scheme, current data is transferred with the previous data to reduce possible chances for re-transmission. When an event from a keyboard or joystick occurs, only control data in the form of 2-byte status information for the current operation as well as additional 2-byte control data for the previous operation is transferred.

The whole 4 bytes do not heavily influence network loads. Once any packet loss or error detected with CRC32 check, the erroneous data is replaced by the control information. In order to raise the efficiency of this scheme, adaptive application to fluctuating packet loss should be

surveyed more.

In the history based re-transmission algorithm (Fig. 6), the number of history stored varies according to a packet loss rate.

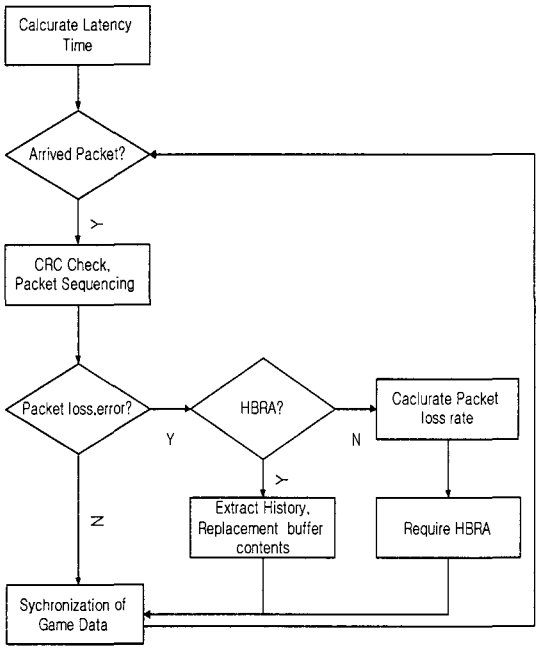


Fig. 6. History Based Re-Transmission Algorithm (HBRA).

4.3 Flow of Network Game Synchronization

The following Fig. 7 shows a conceptual playout synchronized when a data packet from user input arrives. (A) represents the synchronized status when non-history packet transmission is used. Under this scheme, once input events at Xn and Yn occur, the emulator's Callback functions are invoked for a playout.

The transmission delay is decided taking latency time at a player into account. The synchronization delay indicates the delay time required to store data into a synchronization buffer for a playout. If the synchronization delay is not applied, flashes or blinking may be displayed during a playout.

(B) shows a conceptual flow based on a history re-transmission when packet losses are detected. If any packet loss is found at Yn, it could be recovered because a packet having the next

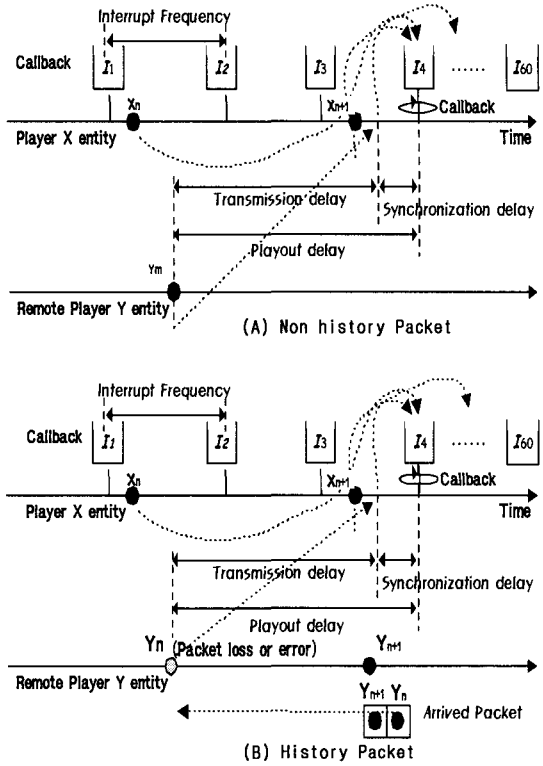


Fig. 7. Flow of Synchronization.

sequence has 2 histories about already arrived packets. Under a UDP transmission, fast synchronization is possible using histories even when packets are not arrived in order.

4.4 Proposed Synchronization Algorithm

The proposed synchronization algorithm is based on the following design considerations to support network status: a relationship between interactive performance and view consistency, an estimation of network status and overhead, and visual synchronization.

A view synchronization scheme has the ability to adjust playout delay time according to network traffic not lowering interactive performance.

The proposed synchronization scheme is organized into 2 stages.

- Stage 1: At this stage, the packet transmission time between each client and a game host is evaluated to calculate latency time in a way that

the time an echo message acknowledged from a client keeping a session as a prompt reply to a message issued from the game host to the client is measured and divided by 2.

The measured transmission time is managed during a game session. A standard latency time is estimated by subtracting transmission time of a client who is to send out a message from the latest transmission time, which can be measured once messages from clients keeping a game session arrive at the game host. When calculating a latency time, waiting time in an event queue should also be included.

Peer-to-Peer network games in general restrict the number of players to 18, but an emulator game is suggested in this paper allows up to 8 players to play with 200ms of the maximum latency time.

■ Stage 2: As a game proceeds, each play measures the status of network traffic at determined intervals to adjust playout latency time according to the traffic status. A sender calculates a packet loss P(i) about packets transmitted to him for ith time interval using the equation (1), and then sends out the result to other players attaching it to a packet originated from the sender.

$$P(i) = (P_{lost}(i) + P_{error}(i)) / P_{total}(i) \quad (1)$$

$P_{total}(i)$ is the number of packets received for i th time interval before actual playout. It is equal to the sum of packet loss(P_{lost}), packet error(P_{error}), packet delay(P_{delay}) and packets received ($P_{success}$).

$$R(i) = Pdelay(i) / Ptotal(i) \quad (2)$$

$R(i)$ is a rate of delayed packets arrived later than playout time. Each player evaluates averages of $P(i)$ and $R(i)$ by extracting the rate of event errors of a sender out of the whole events delivered from other players. When the average of $R(i)$ is greater than that of $P(i)$, each player is notified of a lengthened playout time because it is regarded as heavily loaded status.

In the opposite case, a shortened time is delivered because it is light loaded status. Each player requests individual sender to re-transmit a playout time based on a history algorithm when the value of $P(i)$ of each sender is greater than that of $R(i)$.

4.5 Simulations and Results

The simulation model in this paper assumes 200ms of the maximum delay time and 100ms of the minimum delay time. The rate of packet error is set to 7% with the Poisson distribution. And, 10 packets per second are generated. The number of packets and the rate of delayed packets for the information sent form senders are evaluated every 2 seconds with the varied loads on network traffic in order to calculate an average transmission delay time, which is actually applied to the simulations. When the rate of packet error $P(i)$ is greater than $R(i)$, a history based re-transmission algorithm is applied. In this case, if the rate lies between 5~20%, 2 histories are transmitted.

Fig. 8 shows that the suggested synchronized algorithm represents better performance when the rate of packet errors goes high and, especially in the wireless network environment.

The simulation system is implemented on a Linux server in which a game server designed for this simulation is installed. The game is played in the Windows environment. Fig. 9 is a screen showing a connection with a channel server. Fig. 10 shows a lobby system in which a game room can be created or players are allowed to join in a room.

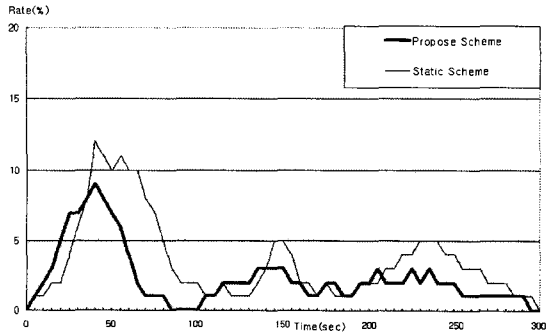


Fig. 8. Packet of History based Re-transmission Algorithm.

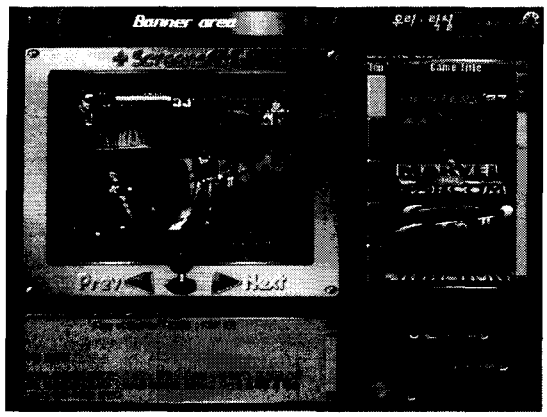


Fig. 9. Channel Server.

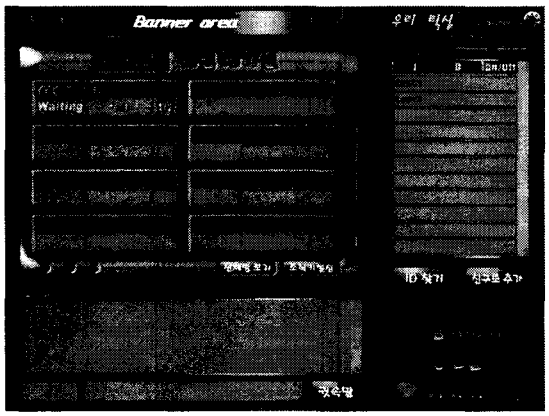


Fig. 10. Lobby system.

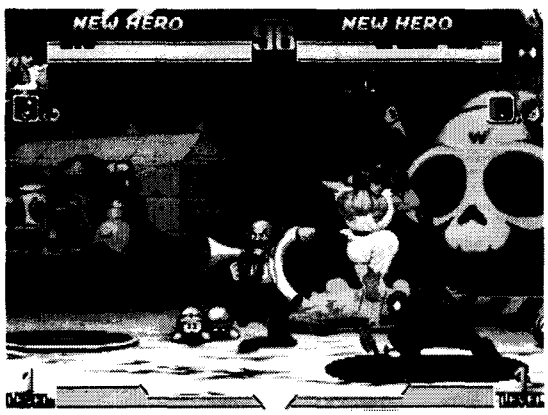


Fig. 11. Play of a Network Game.

5. CONCLUSIONS

In this paper, a distributed server model is designed to make the current single player mode video games be played as network games with a lobby system. The suggested system supports and allows network modules to operate in conjunction with emulators for the current games so that the current games can be enjoyed and played by concurrent multiple players in the form of network games, and based on multi-platform allows ROM Images to be used even without redesigning them, and supports the existing games to be played without any modification.

The suggested system implements an adaptive history re-transmission algorithm for the real-time Peer-to-Peer network games to minimize re-transmission due to packet errors or losses. In case of 3D games, once a packet loss or error occurs, a heavy load on network is inevitably followed by the re-transmission for the packet having the large amount of data even though the packet is compressed. But, the system in this paper only transfer input data from, such as, a keyboard or joystick, to reduce any load on network, and not to affect the load although histories are transmitted. When the adaptive history re-transmission algorithm is applied to the environment of the unreliable UDP transmission, the stability can be secured more so that real-time games are possible by the synchronization acquired by the secured status.

In future, we will study about prediction algorithm for game classification and development of SDK which can support network engine to have general purposes for network games.

6. REFERENCES

[1] On the Geographic Distribution of On-line Game Servers and Players, NetGames, May, pp 173-179, 2003.

[2] Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server, IMW'02, Nov, pp151-156, 2002.

[3] Developing Online Console Games, http://www.gamasutra.com/features/20030328/isen

see_01.shtml.

[4] An Efficient Synchronization Mechanism for Mirrored Game Architecture, NetGames, May, pp 67-73, 2002.

[5] http://www.gbacode.net/

[6] Daniel Bauer and Sean Rooney, Paplo Scotton, "Network Infrastructure for Massively Distributed Games", NetGames 2002, April, 2002.

[7] Tom jehaes and Danny de vleeschauwer, "Access Network Delay in Network Games", NetGames 2003, May, pp63-71, 2003.

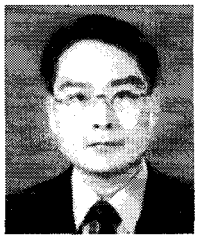
[8] Causality and Media Synchronization Control For Network Multimedia Games, NetGames 2003.

[9] Katherine Guo and Sarit Mukherjee, "A Fair Message Exchanges Framework for Distributed Multi-Player Games," NetGames 2003 May, pp. 29-41, 1998.

[10] L.Gautier and C.Diot, "End-to-end Transmission Control Machanisms for Multipartyr Interactive Application on the Internet", IEEE INFOCOM'99, March 1999, pp. 1470-1479.

[11] L.Gautier and C.Diot, "A Distributed architecture for multiplayer Interactive Application on the Internet", IEEE ICNP'99, July 1999, pp. 6-15.

[12] Matthew K. H. Leung, John C. S. Lui, "Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 9, NO. 6, DECEMBER 2001, pp. 801-817.



Seong-hoo Kim

He is a PhD course graduate student at Kyungnam University studying Network Game and Multimedia system.

Kyoo-seok Park

He is a professor of Computer engineering at Kyungnam University, Korea. He is the honorary president of Korea Multimedia society now. His research focuses on Distributed system, specifically applied to internet computing, Security system and Multimedia system. MS and PhD in Computer science from the Chung-Ang University, Korea.