유·무선 환경에서의 게임명령 히스토리 재전송 기법 기반 실시간 네트워크 게임 시스템☆

A Real Time Network Game System Based on Retransmission Mechanism of Game Command History on Wire/Wireless Environments

김 성 후* 권 영 도** 박 규 석***
Seong-Hoo Kim Young-Do Kweon Kyoo-Seok Park

요 약

본 논문에서는, 다중 플랫폼 기반 비디오 게임을 지원하기 위한 네트워크 게임 시스템을 제안하였다. 제안 시스템은 다수 사용자 게임으로 진행할 수 있도록 설계 하였으며, 실시간 네트워크 게임 진행시 발생되는 네트워크의 부하변동과 지연을 극복할 수 있도록 클라이언트의 초기지연 버퍼링 기법을 이용하여 안정적인 게임진행이 유지되도록 하였다. 또한, 게임 명령의 히스토리 기반 재전송 기법으로 UDP 통신 수단의 패킷 손실 또는 패킷 에러에 대한 단점을 보완하였다.

Abstract

In this paper, we suggest a network game system that can support video game based on multi-platform for multi-user video game. Latency occurring from the load fluctuation in realtime network game is overcomed by using an initial delay buffering scheme on clients, when a real-time game is played, and shows that stable play for a game is achieved as the result of the scheme. Also, We suggest a retransmission algorithm based on the history of game commands, and this Algorithm supplement shortcomings for packet loss and packet error on UDP communication.

☞ Keyword : Network Game, Synchronization, Mutiplayer, Real-time Game, Latency, Buffer, History Packet

1. 서 론

유·무선 환경에서 데스크탑 PC나 PDA, 휴대폰 등의 통합 플랫폼 상에 네트워크 게임을 진행하기 위해서는 여러 가지 제약이 따른다. 게임 하드웨어의 성능과 네트워크 환경에 따른 작용 변수를 가지며 또한, 다수 참가자가 게임을 진행하기 위해서는 각 참가자의 시간지연(Latency)에 대한 적응이 가능해야 한다[1,3,5]. 특히, 실시간 네트워크 게임은 명령 동기화를 위하여 UDP 통신 수단을 사용하여 빠른 응답을 적용하지만, 패킷 손실과 패킷 에러의 문제점을 가지며, 무선 환경에서는 높은 비트 에러와 랜덤 에러로 인해서 여러 개의 패킷손실을 유발하는 특성이 있다[1,10].

실시간 네트워크 게임이 진행되기 위해서는 정확한 게임 데이터가 전송됨으로서 게임 동기화가 이루어져 뷰 동기화에 의한 게임이 진행될 수 있다[8]. 그러나 무선 환경에서는 자주 발생되는 패킷에러와 패킷 손실을 보정하기에는 무리가 따른다. 기존의 게임 동기화를 이루기 위한 방법은 패킷 지연(Delay) 및 패킷 손실을 고려한 AI 보정(Dead reckoning 알고리

* 정 회 원: 경남대학교 컴퓨터공학부 BK21교수
arrayiv@kyungnam.ac.kr
** 정 회 원: 경남대학교 컴퓨터공학부 박사수료
ydkweon@kmail.kimm.re.kr
*** 종신회원: 경남대학교 컴퓨터공학부 교수
kspark@kyungnam.ac.kr
[2006/08/22 투고 - 2006/09/04 심사 - 2006/12/23 심사완료]
☆ 본 연구는 2005학년도 경남대학교 학술진흥연구비 지원에 의해 연구되어졌음.

즘) 기법 및 패킷 스킵을 통한 속도 개선방법 등이 있으나 게임 특성에 따라 적용법이 다르다[8,9].

무선 환경에서 게임을 실시간으로 진행하기 위해서는 패킷 에러 및 손실에 대한 재전송 알고리즘이 필요하며, 재전송 시에도 게임이 실시간으로 진행되기 위해서는 안정적인 버퍼링 기법이 요구된다. 특히, 무선 네트워크를 활용하는 모바일 통신에서 기존의 프로토콜 스택을 수정하게 되면, 오히려 모바일 네트워크 환경이 무거워 지며, 모든 모바일 기기에 디바이스 프로그램을 제공해야하는 단점을 가진다. 본 논문에서 제안하는 재전송 알고리즘은 End-to-End 프로토콜을 지원하여 기존의 프로토콜 변경 없이 재전송 할 수 있다.

본 논문에서는 유·무선 네트워크 환경에서 다중 플랫폼 기반 비디오 게임을 지원할 수 있는 네트워크 게임 시스템을 제안하였다. 이 시스템은 다수 사용자 게임으로 진행할 수 있으며, 실시간 네트워크 게임 진행시 네트워크의 부하변동으로 인하여 발생되는 지연을 극복할 수 있는 클라이언트의 초기지연 버퍼링 기법을 도입하여 안정적인 게임진행이 유지되도록 하였다. 또한, UDP 통신 수단의 패킷 손실 또는 패킷 에러에 대한 단점을 보완할 수 있는 게임 명령의 히스토리 기반 재전송 기법을 적용하였다.

2. 관련 연구

2.1 온라인 게임 데이터의 특징

온라인에서 동작하는 RPG나 슈팅 게임의 경우 게임이 진행되는 동안 사용자의 '이동', '공격'과 같은 명령 데이터는 단순 키 버튼에 의존적이며, 이때 발생하는 이벤트 데이터는 50~130 Byte 정도로, 크기가 작고 빈번하게 집단적으로 발생하는 특성을 보인다.

사용자에게 몰입감을 제공하기 위해서는 일정시간 이내에 명령 전송과 그에 대한 반응 수신이 이루어져야 한다. 사용자는 즉각적인 반응을 얻지 못할 경우 몰입감을 얻지 못한다. 게임 데이터 전송에서 허용되는 Round Trip Latency는 250ms 이내인 것으로 알려져 있다. 게임 데이터의 경우 이 범위 내로 전송 지연을 낮추는 것이 무엇보다 중요하다[2,8,10].

더불어 온라인 게임들은 각각 자신에게 적합한 다양한 수준의 신뢰성 보장과 흐름제어(flow control)를 요구한다. 일부 게임에서는 모든 데이터의 전달이 반드시 보장되어야 하는데 비해 또 다른 게임에서는 데이터 전송 과정에서의 손실이 허용되기도 하고, 혹은 일정 시간이 지나면 데이터 전송이 무의미 해 지는 경우도 있다. 이런 경우에는 손실 데이터에 대해 무한히 재전송하는 것은 불필요하게 네트워크의 자원을 낭비하는 결과를 초래한다.

2.2 모바일 게임 데이터의 특징

무선 이동 통신망이 가지는 고유의 취약점과 무선 단말기의 데이터 처리에 한계점을 극복하기 위하여 모바일 게임 데이터는 단순하게 구성된다. 또한 낮은 데이터 전송속도, 제한된 대여폭, 패킷 전송 실패율, 낮은 연결 신뢰성 등의 제약 사항이 존재하며, 게임 데이터의 전송의 문제점을 해결하기 위해 체크 섬 방법이 사용됨에 따라 재전송에 대한 통신 부담을 준다.

게임 컨텐츠는 아케이드 게임, RPG 게임 등으로 실시간형 또는 턴 형식으로 게임을 진행한다. 아케이드 게임은 I/O 입출력 방식으로 제어된다. 데이터 유형은 키 버튼에 의한 단순 명령으로 구성되어있으며, 보안 적용을 위해 세를러폰 아이디를 암호 또는 키로 이용하기도 한다.

2.3 명령 기반 동기화

명령 기반 동기화 방법은 다음과 같은 순서

로 처리된다. 어떤 사용자가 게임플레이에 어떤 영향을 주었을 때 사용자가 캐릭터를 어떤 장소 또는 방향으로 이동시키거나, 특정 위치로 활을 쏘았을 경우의 행동을 하도록 마우스나 키보드로 입력할 경우, 장치 입력 값 또는 캐릭터 행동에 명령을 주는 값 자체를 메시지화 하여 다른 컴퓨터에 전송한다. 메시지 수신 측은 그 컴퓨터에 있는 캐릭터 객체에 수신된 메시지 행동 명령을 적용시킨다[6][8].

3. 제안시스템

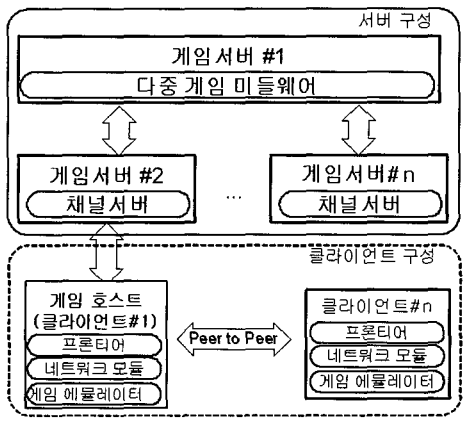
제안 시스템은 네트워크 게임 진행이 가능하며 패킷손실에 대한 재전송 기법의 제공으로 재전송의 효율성을 높여 모바일 기기에 적용시 패킷의 수를 줄일 수 있다.

본 시스템은 실시간 P2P 진행방식이며, 네트워크 게임 동기화는 키보드나 조이스틱에서 발생되는 2바이트의 상태 레지스터를 이용하여 상대방의 입력포트에 전송하여 동작되도록 한다.

두 플레이어는 동일한 화면으로 조이스틱을 이용하여 인터넷상에서 원격으로 조정하며, 뷰 동기화 표현을 위해 초기 버퍼링 지연방법으로 네트워크 게임을 진행한다. 모바일 기기간의 게임은 게임서버가 중계역활을 담당하고, 모바일 기기와 PC간의 게임진행은 PC가 중계역활을 담당하며, PC간의 게임 진행은 게임 개설자가 담당한다.

3.1 분산 네트워크 게임 시스템 구조

<그림 1>은 제안 네트워크 게임 시스템의 구조이다. 미들웨어는 게임 서버를 중재하는 역할로서 클라이언트와 TCP 컨넥션으로 정보교환과 세션연결을 처리하며, 플레이어가 접속하면 미들웨어에 의해 분산처리 되어 접속하게 된다. 채널서버는 게임선택에 따라 종류별로 배정한다.



〈그림 1〉 분산 네트워크 게임 시스템 구조

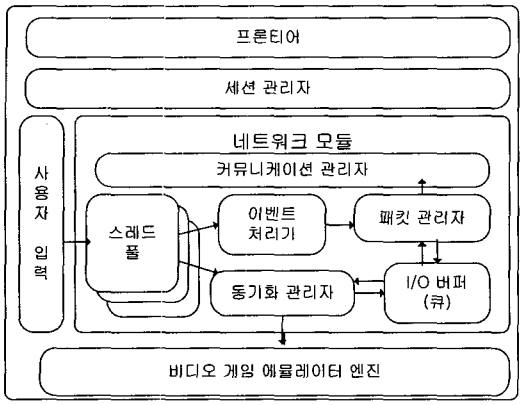
게임서버에서 각각의 클라이언트가 게임 룸을 생성하면, 그 클라이언트는 호스트 컴퓨터로 지정되어 게임 룸(ROOM)을 관리하며, 모바일기기간의 게임 진행시에는 채널서버가 게이트웨이 서버 기능을 한다.

3.2 네트워크 게임 클라이언트

게임 클라이언트 구조는 프론티어, 세션 관리자, 네트워크 모듈, 비디오 게임 엔진으로 구성된다. 네트워크 모듈은 쓰레드 단위로 사용자 입력처리와 패킷 생성 및 패킷 분석을 처리하며, 동기화 관리자를 둔다. 패킷 관리자를 통해 사용자의 입력 내용과 출력 내용을 I/O 버퍼로 저장하며, 패킷 손실 및 유실을 측정하여 패킷에 대한 재전송 및 지연시간을 결정한다.

동기화 관리자는 I/O 버퍼에 있는 순서화 된 정보를 에뮬레이터의 Callback 함수로 입력 내용을 게임 엔진에 동기화 시킨다. 통신 관리자는 참가자들 간 통신 채널의 생성, 삭제 등을 수행하며 시스템 구조에 따라 P2P 구조를 지원한다.

세션관리자는 참가자의 정보를 관리하며, 최초 연결시의 지연시간을 검출하여 네트워크 게임 동기화에 적용한다.



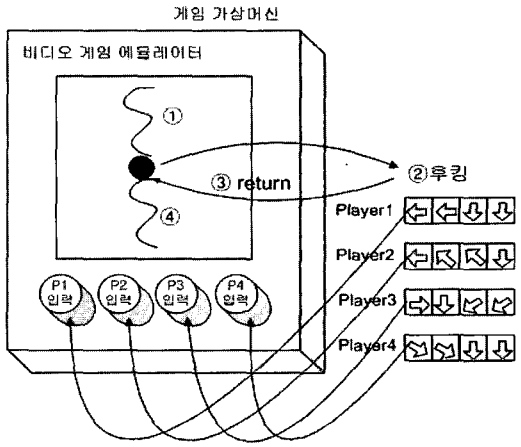
〈그림 2〉 게임 클라이언트 구조

모바일기기와 PC간의 게임 진행시 클라이언트가 게임진행을 관리하는 서버로서 활동하며 1:1 진행이 지원된다.

3.3 게임 엔진의 I/O 입력 맵핑 처리

<그림 3>은 타임스탬프 간격으로 입력된 입력 커맨드 데이터와 전송 받은 입력 데이터를 버퍼에 저장하는 방법을 나타내며, 버퍼 내용은 게임 에뮬레이터에 적용된다. 버퍼 내용은 순차적으로 Sync 번호에 의해 저장되며, 키보드나 조이패드의 상태 값을 저장한다. 조이패드나 키보드의 입력이 없는 상태는 명령이 없는 것으로 처리되지만, 동기화 진행을 위해서는 순차적인 커맨드 데이터를 보유하게 된다. 타임스탬프시간에 버퍼의 값은 게임 에뮬레이터의 하드웨어 버튼의 상태 값으로 전달되어 작동된다.

명령처리 순서는 ① 게임진행시 초당 30 프레임 또는 60 프레임 간격으로 진행 되어지며, ② 화면 주사시 인터럽트 후킹 수행시에는 프레임 진행을 멈추게 되며, ③ 버퍼에 저장된 각 플레이어의 커맨드 데이터는 할당된 조이패드로 맵핑되며, ④ 인터럽트를 return시켜 버퍼에 저장되어 있는 커맨드 데이터가 에뮬레이터에 의해 하드웨어의 버튼 상태 값으로 적용되어 게임이 진행 된다.



〈그림 3〉 플레이어의 입력된 버퍼의 명령 처리도

3.4 게임 데이터의 구조

게임명령 히스토리에 기반한 재전송 알고리즘을 구축하기 위한 프로토콜을 다음과 같이 설계하였다. 패킷은 길이(Length), 헤더(Header), 플레이어(Player), 동기화번호(Sync Num), 데이터로 구성하였으며, Length는 패킷 총 길이이며, Header는 컨트롤 데이터 타입을 가진다. 패킷 데이터 타입과 구조는 <그림 4>와 같다.

8bit	8bit	4bit	4bit	가변길이	16bit
Length	Header	Player	Sync Num	Data	엔드마커

〈그림 4〉 게임데이터 구조

헤더는 Control 데이터, Infomation 데이터, N Based 히스토리 데이터, 재전송 요청 + N Based 히스토리데이터로 구성된다.

<그림 5>는 3개의 게임 데이터로 구성된 히스토리 패킷구조를 나타낸 것이다.

3 based history packet(PK_3HISTORY_DATA)

8bit	8bit	8bit	8bit
Length	Header	Player	Sync Num
Input Data 3		Input Data 2	
Input Data 1		End Mark	

〈그림 5〉 히스토리 게임 데이터

3.5 플랫폼간 유무선 연동 기술

제안 시스템에서의 유무선 연동기술은 무선 vs 유선 또는 무선 vs 무선 두 가지 형태로 구성된다. 무선인 경우는 모바일기기 또는 PDA 대상이다. 유무선 연동기술은 다중 플랫폼에서 동시에 플레이 되는 게임이기 때문에 모든 플랫폼에 대해서 동일성을 보장해야한다. 이를 보장하기 위하여 상호간의 프레임 비율과 네트워크 속도를 고려하여 버퍼 크기와 타임스탬프를 결정하게 된다.

모바일 기기와 PC간의 게임 진행은 입력장치 인터페이스가 다르기 때문에 값 변환 테이블을 가지고 있으며, 유선 패킷과 무선 패킷을 구분하여 변환한다. 무선 패킷이 구성될 때 Parity bit를 추가하여 무선망의 패킷 손상 유무를 알 수 있도록 구성하였으며, '무선 클라이언트-패킷변환-유선 클라이언트(게임진행 서버)' 형식으로 패킷을 전송한다.

모바일기기간의 게임 진행시는 채널서버가 관리 하며 모바일 기기간의 통신을 전달하는 게이트웨이 서버 기능을 수행한다. 본 실험에서는 1:1로 게임을 진행하며, '무선 클라이언트-게이트웨이 서버(게임진행 서버)-무선 클라이언트' 형식으로 패킷을 전송한다.

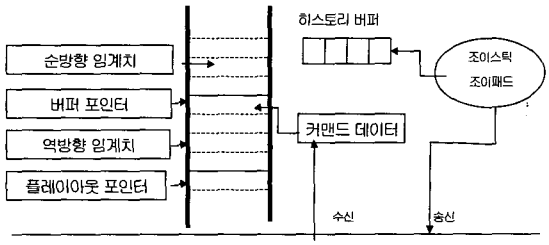
4. 네트워크 게임 전송 시스템의 구성

본 논문에서 제안하는 게임 클라이언트는 게임의 안정적인 진행을 위하여 각 클라이언트는 완충역할을 수행하는 버퍼를 가진다. 네트워크 지터 현상과 UDP 방식의 전송으로 패킷손실 또는 패킷 에러를 방지하기 위하여 전송된 패킷에 대한 손실여부를 체크해야 하며, 손실된 패킷은 히스토리 재전송 요청을 통해 버퍼를 재배치시킨다. 지터의 발생 시 최대 2배수 수준까지 버퍼크기를 정하여 부하변동에 적응할 수 있도록 하였다.

4.1 버퍼 관리

<그림 6>에서 역방향 임계치는 스타베이션 경고구간으로서 스타베이션이 발생되기 전에 각 플레이어들에게 알릴 수 있는 타임구간이며, 순방향 임계치는 오버런이 발생되기 전에 각 플레이어들에게 알릴 수 있는 타임 구간이다. 발생 신호를 받은 플레이어들은 일시적으로 수행을 멈추게 되며, 다시 버퍼링 지연시간이 지나 게임이 진행된다.

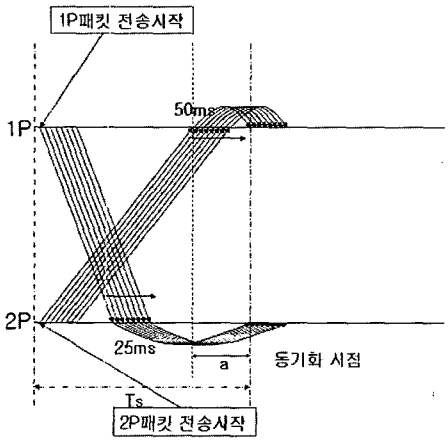
게임 클라이언트에서 버퍼관리는 참가한 플레이어 수만큼 독립적인 버퍼가 존재하며, 키보드나 조이스틱의 신호를 받아 히스토리 버퍼에 저장한다. 이는 재전송 요청시 이용된다.



〈그림 6〉 버퍼관리 구조

<그림 7>에서 버퍼 구조는 T_S만큼의 버퍼 값을 초기 설정하고, T_S시간 이후부터는 일종의 데이터 스트림처럼 키 입력을 처리하되 동기화를 위하여 타임스탬프의 적절한 값을 구해야 한다. 두 플레이어간의 패킷 전송시간 중

느린 쪽의 전송 시간에 일정한 시간을 추가하여, 초기 a의 추가 버퍼 값을 일정한 공식에 의하여 구해야 하며, 최초로 T_S값의 버퍼 값이 소진된 시점으로부터 다이내믹하게 T_S의 값을 변경시켜 나갈 필요가 있다.



〈그림 7〉 Input/Output 데이터 동기화

0.1 sec ~ 0.3 sec 까지는 Key 입력이 화면에 늦게 반영된다 하더라도 게임을 진행하는데 큰 무리가 없으며, 실제로 플레이어는 지연 차이를 전혀 인식하지 못한다. 버퍼는 0.3 sec 동안의 초기지연에 의해 커맨드 데이터를 확보하고 입력은 출력의 0.3 sec 지연시켜 게임진행을 하게 된다.

타임스탬프는 식 1과 같이 정의 된다.

(RTT_{max} : Maximum of Round Trip Time, RTT_{min} : Minimum of Round Trip Time,FPS : Frame Per Second)

$$Ts = Max((RTTmax/2 - RTTmin/2), 1/FPS) \quad (1)$$

타임스탬프는 게임기의 프레임 처리율과 네트워크 지연성에 따라 간격이 달라지며, 네트워크 성능이 우수하다면 최대 프레임 처리가 가능하다.

4.2 버퍼 크기 결정

본 연구에서 버퍼 크기는 게임 진행시 안정적인 게임 진행과 최적의 동기화가 되도록 설정하였으며, 1:1 게임 진행시 버퍼 크기 설정과 서로 다른 네트워크 상황을 가지는 다수 플레이어의 버퍼 크기를 가지도록 하였다.

버퍼 크기 결정 시기는 게임 진행 초기에 각 플레이어들의 프레임 처리율, RTT, 지터 허용 수치에 의해 결정된다. 이후에는 스타베이션 발생시 또는 오버런 발생시 그리고 플레이어의 게임 진행시 탈퇴되는 경우, 버퍼 크기를 재설정하게 된다.

1:1인 경우, 초기 버퍼 크기는 식 2에 의해 결정한다.

$$\Delta B_{size} = B_{delay} + RTT_{\max} * T_S \quad (2)$$

동기화 재설정시, 식 3과 같이 버퍼 크기가 결정된다.

$$\Delta B_{size} = \frac{B_{delay} - Max(RTT/2)}{Max(RTT)/2 - Min(RTT)/2} \quad (3)$$

3인 이상 참가시 식 4와 같이 버퍼 크기가 결정된다.

$$\Delta B_{size} = \frac{B_{delay} - Max(RTT/2)}{Max(RTT)/2 - Min(RTT)/2} \quad (4)$$

3인 이상 참가시 동기화 재설정시, 버퍼 크기는 식 5에 의해 결정된다.

($\Delta^c_{\max}$: maximum latency of arrival command, $\Delta^c_{\min}$: minimum latency of arrival command)

$$\Delta B_{size} = \frac{B_{delay} - \Delta^c_{\max}}{\Delta^c_{\max} - \Delta^c_{\min}} \quad (5)$$

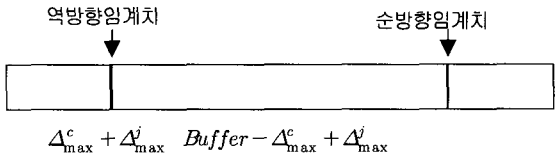
버퍼 크기는 네트워크 성능에 따라 변할 수 있으며, RTT 값에 따라 타임스템프를 결정한다. FPS는 게임기의 출력 상태에 따라 결정한다. 1:1 재동기화 과정은 RTT 값을 고려하여 결정하지만, 3인 이상일 때는 현재 커맨드 데이터 도착시간의 차이에 따라 재동기화 한다.

4.3 버퍼값의 임계치 구간

게임 클라이언트 버퍼의 역방향 임계치 구간은 버퍼에 저장되어 있는 커맨드 데이터가 고갈되기 전에 이상현상에 대해 각 플레이어에게 통보하여 버퍼 지연처리를 요청하여 버퍼를 다시 채우는 경우와, 3인 이상이 게임 진행시 게임 클라이언트에 있는 상대편의 버퍼를 검사하여 서로 다른 버퍼 크기를 가지는 경우, 재 동기화 요청을 하게 되는 두 가지 방법으로 수행한다.

이러한 역할을 수행하는 임계치 구간은 $\Delta^{c}_{max} + \Delta^{j}_{max}$ 시간의 여유를 확보해야 자원이 고갈되기 전에 각 플레이어에 대한 버퍼 지연처리 또는 재동기화가 가능하다.

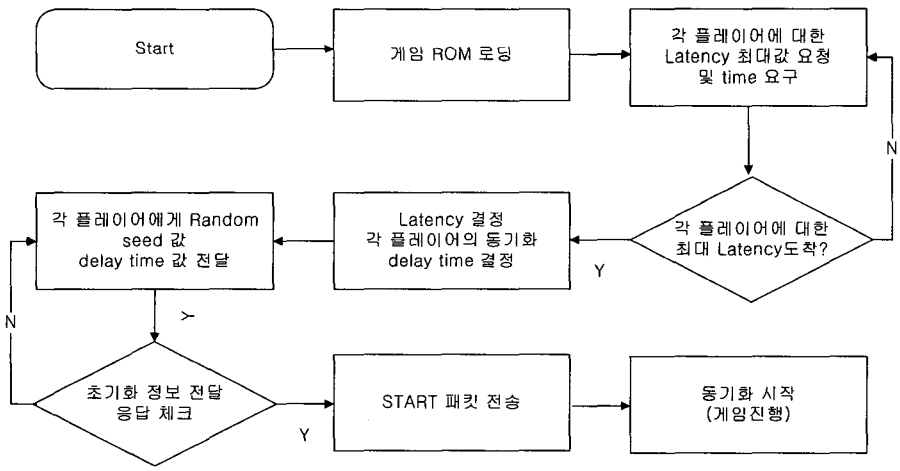
(Δ^{j}_{max}: maximum jitter of arrival command, Δ^{c}_{max}: maximum latency of arrival command)



〈그림 8〉 임계치 구간

4.4 네트워크 게임 Start 동기화

P2P 방식의 네트워크 플레이어 그룹이 완성되면, <그림 9>와 같이 에뮬레이터 게임을 수행하기 위한 Start 동기화 과정을 진행한다. 이 과정이 수행 되어야 실시간 동기화 데이터에 대해 최대한의 지연성(Latency)을 극복할 수 있다. 본 논문에서 제안한 멀티플레이어 수 는 최대한 많은 인원이 수행할 수 있도록 하였으며, 서로 다른 지연성을 가질 경우에도 같은 시간대 수행이 가능하다.

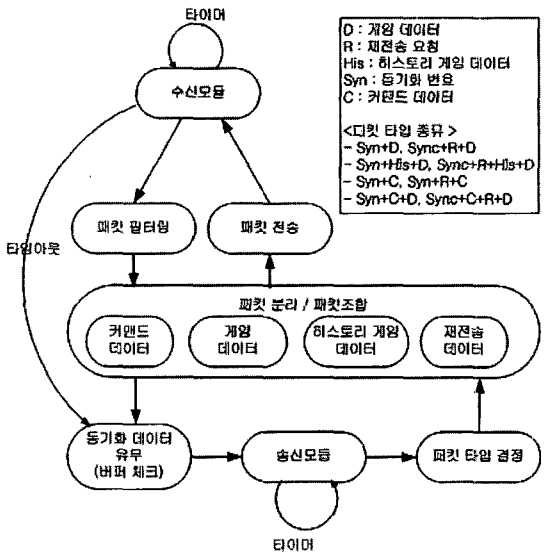


〈그림 9〉 네트워크 게임 Start 동기화 수행도

각 플레이어에 대한 지연시간을 구하기 위해 PING 정보로 각 플레이어의 평균 RTT를 구하여 게임 진행 호스트에 대해 최대 지연시간을 전송하게 된다. 게임 진행 호스트는 각 플레이어에 대한 최대 지연시간과 자신이 체크한 각 플레이어의 지연시간 정보를 이용하여 딜레이 시간을 결정한다. 결정된 딜레이시간과 랜덤 seed 값을 각각의 플레이어에게 전달하여 게임을 진행하도록 한다. 랜덤 seed 는 게임의 랜덤 발생 초기 값으로 활용되며, 각 플레이어가 같은 게임 시나리오로 진행될 수 있도록 하는 값이다. 딜레이 시간은 각 플레이어에 대해 동시 진행을 할 수 있도록 유도하는 게임 진행 지연시간이다. 게임 진행자가(최대 지연시간 - 플레이어 지연시간) 계산하여 각각의 플레이어에게 딜레이 시간을 전송하며, 전송받은 플레이어는 결정되어진 딜레이 시간만큼 기다렸다가 게임을 진행한다.

4.5 히스토리 재전송 알고리즘

히스토리 재전송 알고리즘은 커맨드 수신부와 커맨드 송신부로 구성되며, 양방향으로의 데이터 전송을 위해서 여러개의 타이머를 사용한다. 커맨드 수신부는 게임 데이터를 분석한 후, 게임 데이터 인지 게임 명령인지를 인지한다. 게임 데이터이면 버퍼에 배치하며, 게임 명령이면 제어를 따르게 된다. 게임 데이터이면 동기화 번호(Sync Num)를 조사하여 잃어버린 게임 데이터가 있는지를 판단한다. 동기번호가 순차적이지 않을 때, 잃어버린 게임데이터 동기번호를 기억하였다가 자신이 게임 데이터를 보낼 때 함께 동기번호의 게임데이터와 요청 메시지를 재전송 요청한다. 상대편의 커맨드 수신부는 게임 데이터가 수신되었을 때, 재전송에 대한 정보를 기록하였다가, 재전송 정보를 게임데이터 전송시 포함하여 전송한다. 이때, 상대편의 동기번호가 순차적이지 않으면 재전송 요청 메시지를 게임 데이터와 함께 보낸다. 그림 10은 재전송 알고리즘을 나타낸 것이다.



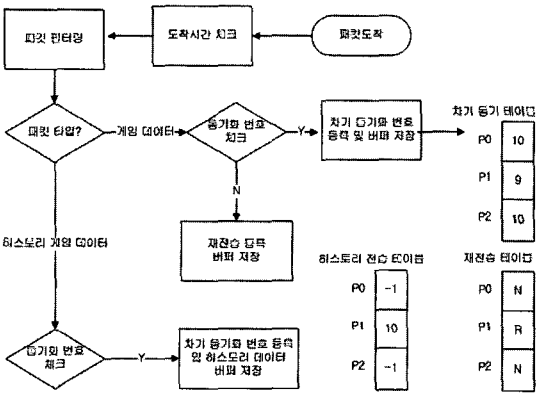
〈그림 10〉 히스토리 재전송 알고리즘

연속적인 패킷 손실이 발생될 때, 플레이어는 수신 데이터 보다 버퍼내 데이터의 출력이 빠르기 때문에 스타베이션 현상이 발생하게 된다. 이 경우, 재동기화 게임 명령어를 보내어 게임 진행을 멈추게 하고, 재동기화를 위한 start 동기화 과정을 거치게 된다.

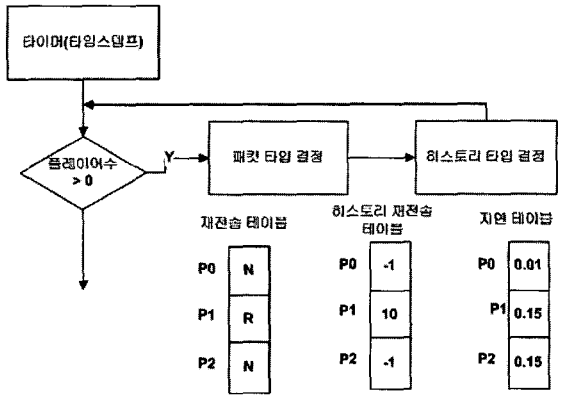
4.5.1 히스토리 재전송 수신부

재전송을 요청하기 위해서는 전송받은 패킷을 필터링하여 동기화 시퀀스 넘버와 커맨드 데이터를 분리한다. 이 경우, 타임스탬프 기준으로 패킷 재전송 요청을 결정하며, 재전송 요청시 송신측에서 자신의 게임 데이터 및 재전송요청 메시지를 포함하여 전송한다. 수신측에서 재전송 요청 패킷을 받게 되면 보낼 데이터에 재전송 요청된 게임 데이터를 포함하여 전송하게 된다.

<그림 11>은 히스토리 재전송 알고리즘의 수



〈그림 11〉 히스토리 재전송 알고리즘의 수신 처리도



〈그림 12〉 히스토리 재전송 알고리즘의 송신 처리도

신 처리도이며, 패킷이 도착되면 패킷 필터링에 의해서 메시지 타입이 분류된다. 분석된 메시지가 재전송 히스토리 데이터인 경우, 길이만큼 버퍼에 채우게 되고, 일반 데이터인 경우, 패킷 손실에 대비하여 차기 동기 테이블을 두어 현재 수신된 동기화 번호의 다음 번호를 마킹한다. 만약, 현재 들어온 동기화 번호와 예측되는 동기화 번호가 다르다면, 재전송 테이블에 'R'을 표시하여 자신이 보낼 데이터에 재전송 요청 메시지를 포함하도록 지시한다. 히스토리 전송 테이블은 송신자에 의해 요구된 동기화 번호의 기록들이다.

4.5.2 히스토리 재전송 송신부

<그림 12>는 히스토리 재전송 송신 처리도이며, 타임스템프 간격으로 작동한다. 자신이 송신할 데이터는 재전송 테이블, 히스토리 전송 테이블에 의해 송신할 패킷 타입과 히스토리 데이터의 크기가 결정되며, 수신하지 못한 동기화 데이터 번호가 있을 경우 재전송 테이블 정보를 이용하여 재전송 요청 메시지 패킷을 생성한다.

4.6 모바일 인터넷을 위한 오류 보정

무선 인터넷에서 패킷 손실 또는 에러에 대한 보정 기법은 다음과 같다.

(1) 연속적인 에러에 대비한 기법
무선 인터넷에서 무선 기지국 또는 무선 AP와 거리가 멀어질 때, 에러가 연속적으로 자주 발생한다. 연속적인 에러가 발생할 때, 에러에 대한 가중치를 설정하여 2-Based History, 3-Based History 방식의 게임 데이터 전송을 요청한다. 이 방식은 게임 데이터 전송 시, 2byte, 4byte 게임 데이터만 늘어나며, 재동기화 과정이 줄어든다.

(2) 프레임 대비 재전송 스킵 기법
모바일 기기는 초당 6프레임을 출력할 수 있는 성능을 가지고 있으며, PDA는 10 프레임을 출력할 수 있다. 입출력을 위한 인터럽트가 많이 발생할수록 CPU 점유률이 높아져, 프로세싱 타임이 현저히 떨어진다. 모바일 기기에서 게임명령은 키보드 입력 처리속도에 따라 게임이 진행된다.
Sync Num-1 번째 게임데이터가 손실되었을 때, (Sync Num-2), (Sync Num) 번째의 게임 데이터를 정상적으로 가지고 있다면, 두개를 비교하여 동일한 경우, Syn Num-1 번째를 동일한 명령어로 처리한다. 만약, 다르다면 재전송을 요청해야한다.

5. 구현 및 평가

5.1 실험환경

본 논문에서 제안한 네트워크 엔진의 성능을 평가하기 위하여 Ethereal을 사용하여 트래픽을 분석하였으며, 비교분석을 위해 M사의 스트리트 파이터 게임과 공개 에뮬레이터인 MAME를 사용한 스트리트 파이터를 이용하였다.

성능 평가는 제안된 시스템과 M사의 엔진, MAME의 게임을 각각 60초씩 수행하면서 Packet/sec (초당 전송되는 패킷 수), 프로토콜 비교, FPS(초당 교체되는 프레임) 수를 비교 측정하였다.

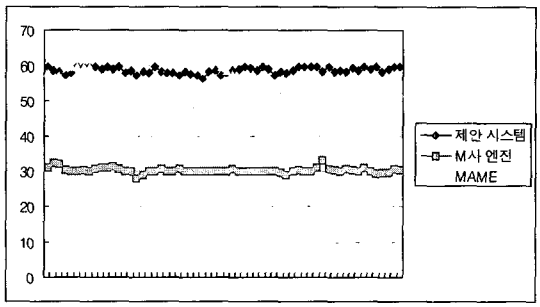
5.2 실험결과 및 평가

M사는 TCP 프로토콜을 사용하여 프레임 처리율이 고정적이며, 지터가 발생할시 전체적으로 게임 진행이 느려지는 현상이 발생하며, ACK 패킷의 전송으로 오버헤드가 많이 발생되는 문제가 있음을 알 수 있다. 또한, MAME는 UDP를 사용하며, 체크섬을 통한 패킷보정 방법을 사용함으로써 복구하지 못한 패킷은 스킵 방식으로 처리되어 프레임 처리율이 유동적임을 알 수 있다. 패킷전송 결과에 대한 비교 내용은 <표 1>과 같으며, 네트워크 엔진 프로토콜의 비교내용은 <표 2>와 같다.

〈표 1〉 패킷전송 비교

	제안 시스템	M사 엔진	MAME
평균 packet / sec	106.519	112.248	114.762
평균 packet 크기(byte)	62.621	67.245	72.274
평균 byte / sec	7364.889	7548.117	8523.833

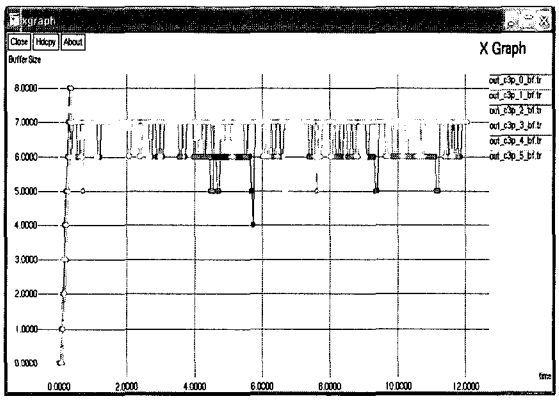
<그림 13>은 1:1 플레이어간의 최대 전송 알고리즘을 적용시켜 FPS를 측정한 결과를 나타낸 것이다. 여기에서 제안된 시스템과 MAME가 전체적으로 60 FPS의 우수한 수준을 유지하고 있음을 알 수 있으며, MAME의 경우 체크섬 방법으로 게임데이터를 복구함으로서 FPS의 수가 일정하지 않아 안정되지 않음을 알 수 있다. 그리고, M사의 경우 비디오게임의 부드러운 동영상을 일정부분 포기하고 30 FPS에 목표를 고정하고 있음을 알 수 있다.



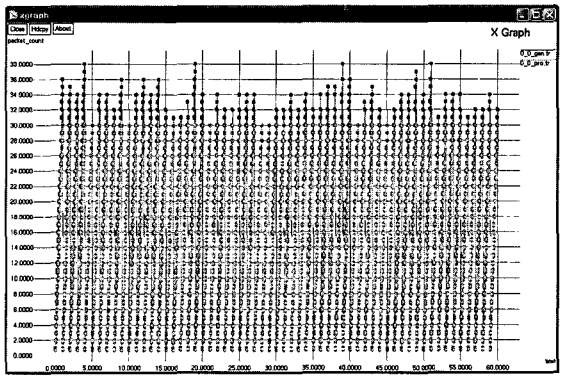
〈그림 13〉 1:1 플레이어간의 FPS 성능

〈표 2〉 네트워크 엔진 프로토콜 비교

	제안 시스템	M사 엔진	MAME
프로토콜	UDP	TCP	UDP
통신방식	Peer to Peer	Server - Client	Peer to Peer
타임스템프	사용	비사용	사용
체크섬	사용	비사용	사용
패킷손실 보정	History 명령어 기반 재전송	ACK/NACK	에러 체크 및 Skip
프레임 당 Data Byte	6~10 byte	16byte	32byte



〈그림 14〉 3% error 적용시 버퍼크기



〈그림 15〉 error 5%적용시 제안 알고리즘과 재전송 패킷 수 비교

5.2 NS2 시뮬레이션 결과

제안 알고리즘에 대해 NS2 시뮬레이션으로 효율성을 분석하였다. <그림 14>는 패킷 에러 손실을 3% 적용시, 버퍼 내용이 채워져 있음을 알 수 있으며, 0.3초간의 초기지연 시간을 주어서 버퍼 내용이 채워진 상태에서 게임 진행으로 버퍼가 소진됨을 알 수 있다.

<그림 15>는 패킷 에러 손실 발생시, 패킷 손실에 대한 일반 재전송 요청방식과 제안 알고리즘의 경우를 비교한 결과이다(패킷 손실률은 5%를 적용하였다). 제안 알고리즘은 패킷 손실에 대하여 패킷수가 일률적으로 발생되지만, 일반 재전송 방식은 패킷수가 늘어남을 알 수 있다.

6. 결 론

제안 알고리즘은 패킷손실을 체크할 수 있는 프레임에 대한 동기화 번호를 사용하였으며, 연속적인 패킷손실을 방지하기 위해 타임스템프에의 의해 체크 할 수 있도록 하였다. 게임 진행시 동일한 게임 뷰를 유지하기 위해 상호 지연시간을 고려하여 타임스템프 간격에 따라 게임 명령 패킷 전송이 이루어지도록 하였으며, 손실된 패킷을 보완하는 게임 명령의 히스토리 기반 재전송 알고리즘은 어플리케이션 프로토콜로 설계 하였다.

제안 시스템은 메시지 송·수신시, 재전송 요청과 게임 진행을 위한 제어코드와 게임 명령 데이터를 전송하여, 패킷 손실에 대한 재전송 요청시 요구된 게임 명령 데이터를 포함하여 전송함으로써 재전송에 따른 추가 비용이 발생되지 않으며, 히스토리 방식의 재전송 알고리즘의 구현으로 패킷 손실 또는 유실에 따른 재전송 오버헤드를 최소화하여, 참여자간에 실시간 게임을 가능케 한다.

향후 네트워크 성능이 우수한 게임 클라이언트가 중심이 되어 각각의 클라이언트에게 재전송 중계가 가능한 메카니즘에 대한 연구가 필요하다.

참 고 문 헌

[1] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," in Proc. Workshop on Network and System Support for Games, Aug. 2004, pp. 144 - 151.

[2] Wu-chang Feng, Francis Chang, Wu-chi Feng, Jonathan Walpole, "A Traffic Analysis of a Busy Counter-Strike Server", IMW2002, Nov, pp151-156, 2002.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V Jacobson, "RTP: A Transport Protocol for Real-time Applications," RFC 1889, Jan. 1996.

[4] J. Färber, "Network Game Traffic Modelling", Proceedings of NetGames 2002, pp. 74-78, Braunschweig (BRD), 16-17 April 2002.

[5] 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.

[6] Eric Cronin, Burton Filstrup, Anthony R. Kurc, Sugih Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architecture", NetGames, May, pp 67-73, 2002.

[7] G. Armitage. "Sensitivity of Quake3 players to network latency". ACM SIGCOMM Internet Measurement Workshop 2001, Berkeley, CA, USA, Nov. 2001.

[8] N. Baughman and B. Levine. "cheat-proof playout for centralized and distributed online games". In Proc. Infocom 2001, pp. April 2001.

[9] Cai, W.,Lee, Francis B.S., "Auto-Adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation". Proceedings of Thirteen Workshop on Parallel and Distributed Simulation, pp.82-89, 1999.

[10] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," in Proc. Workshop on Network and System Support for Games, Aug. 2004, pp. 144 - 151.

[11] 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.

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

[13] T. Bova and T. Krivoruchk, "Reliable UDP protocol," Internet Draft, draft-ietf-sigtran-reliable-udp-00.txt.

[14] Diot, C. and Gautier, L., "A Distributed Architecture for Multiplayer Applications on the Internet", IEEE Networks Magazine, Vol. 13, NO. 4, July-August, 1999.

◐ 저 자 소 개 ◑

김 성 후 (Seong-Hoo Kim)
1995년 경남대학교 전산통계학과 졸업(학사)
1997년 경남대학교 대학원 전자계산학과 졸업(석사)
2005년 경남대학교 대학원 컴퓨터공학과 졸업(박사)
1998년~2004년 창신대학 조교수
2005년 정보통신진흥원 IT교수 요원
2006년~현재 경남대학교 컴퓨터공학부 BK21교수
관심분야 : 멀티미디어 네트워크게임, u게임, 지능형 홈네트워크
E-mail : arrayiv@kyungnam.ac.kr

권 영 도 (Young-Do Kweon)
1993년 경남대학교 전자계산학과 졸업(학사)
1995년 경남대학교 대학원 전자계산학과 졸업(석사)
2000년~현재 경남대학교 대학원 컴퓨터공학과 박사수료
1995년~현재 한국기계연구원 연구기획실 선임연구원
2005년~현재 경남 U-포럼 운영위원
관심분야 : 그리드컴퓨팅, 유비쿼터스 컴퓨팅, 네트워크 게임
E-mail : ydkweon@kmail.kimm.re.kr

박 규 석 (Kyoo-Seok Park)
1971년 계명대학교 경영학과 졸업(학사)
1981년 중앙대학교 대학원 전자계산학과 졸업(석사)
1988년 중앙대학교 대학원 전자계산학과 졸업(박사)
1990년 UCLA 컴퓨터공학과 객원교수
1982년~현재 경남대학교 컴퓨터공학부 교수
관심분야 : 시스템소프트웨어, 홈네트워크, 멀티미디어 시스템, CRM
E-mail : kspark@kyungnam.ac.kr