HeartBeat을 이용한 고가용성(High Availability) 보장 온라인 게임 서버 구축 제안

Research About Design and Implementation of On-Line Game Server for High Availability guarantee using Heartbeat

김태열, 경병표*, 유석호, 이완복
공주대학교, 중부대학교*

Kim Tae-Yul, Kyung Byung-Pyo*, Ryu Seuc-Ho, Lee Wan-Bok
Kongju National Univ., Joongbu Univ.*

요약

인터넷을 이용한 서비스와 산업 그 중에서 게임 산업의 급속한 발전으로 인하여 게임 서비스를 더욱 효율적이고 확장성이 뛰어난 시스템으로의 구축이 중요시되고 있다. 이에 고가용성 시스템이 주목을 받고 있고 본 논문에서는 heartbeat 알고리즘을 이용한 고가용성 온라인 게임 서버를 제안한다. 분산화된 게임 서버에 대하여 각 대응되는 standby server를 두어 실시간 real server의 장애를 체크하여 지속적인 게임 서비스에 지장이 없도록 하여 항상 안정된 게임 서비스를 할 수 있는 시스템이다.

Abstract

Providing stable services has been a crucial factor to determine success or failure in current situations, in which online game industry has become one of the important industries. The study proposed a method of establishing a stable game servers using 'Hearbeat', one of the existing (load) balance server techniques for the enhancement of availability of online game servers. In the proposed method, one could provide users with stable game services, by inspecting master server frequently using stand-by servers corresponding to each game server.

I. 서 론

1. 연구의 필요성

20세기 현재의 한국의 게임 산업의 대표는 온라인 게임이다. 컴퓨터 성능의 비약적인 발전과 함께 초고속 네트워크의 빠른 보급으로 대표적인 온라인 게임 선진국이 되었다. 온라인 게임은 대규모의 사용자들이 동시에 접속하고 동시 접속한 사용자들은 채팅(Chatting)과 함께 게임을 즐긴다. 이러한 영향으로 게임 서버에는 많은 Traffic이 오가면서 게임 서버의 성능에 많은 영향을 끼치게 된다. 수많은 사용자들의 접속으로 인하여 게임 서버의 부하는 점점 더 심해질 것이고 그에 따라 게임 서비스는 계속적으로 느려질 것이다. 심지어는 게임 서버의 과부하로 인하여 시스템이 다운되는 사태까지 발생할 수도 있다. 시스템의 과부하 및 다운으로 인한 게임 서비스의 장애는 사용자로 하여금 게임에 대한 신뢰성에 대한 문제가 될 수 있다. 게임 서비스의 장애는 게임 사용자로부터 게임에 대한 신뢰성을 잃게 되는 원인이 되며 이것은 궁극적으로 게임 업체의 이익에 중대한 영향을 미치게 된다. 게임업체들은 시스템의 과부하 및 게임 서버의 다운을 방지하기 위한 여러 가지 방법을 선택하여 게임 서버에 적용하고 있다. 하지만 게임 서버에 적용한 다양한 기술 또한 궁극적으로 게임 서버의 과부하 및 서버 다운에 대한 문제 해결방안은 아니라 할 수 있다.

경쟁화 되어가는 게임 업계들은 온라인 게임 서비스 분야에서 우위를 가지기 위해서는 안정된 서비스를 통하여 게임 사용자들의 신뢰성 확보가 시급하다고 본다. 신뢰성 확보를 위해서는 게임 서버의 가용성을 높여야 한다. 게임 서버의 고가용성을 통하여 게임 사용자들에게 질 높은 게임 서비스를 제공할 수 있을 것이다.

게임 서버의 고가용성은 게임 사용자와 게임 업계 모두 만족할 만한 선택이며, 사용자의 신뢰도 및 게임의 충성도를 높이는데 없어서는 안 될 중요한 기술이다.

2. 연구의 방법 및 내용

본 연구는 온라인 게임 서버의 지속적이고 안정적인 서비스를 위하여 하나의 방법으로써 Heartbeat을 이용하여 분산된

게임 서버들의 장애 및 서버 다운현상에 빠르게 대처할 수 있는 고가용성 온라인 게임 서버 시스템을 구축하기 위한 방법의 제안하는 것을 목표로 한다.

본 논문에서는 이론적인 내용과 실제적인 시스템 구현을 위한 시스템 제안의 두 부분으로 구성되어 있다. Ⅰ장의 서론부분에서는 연구의 필요성, 그리고 연구 방법 및 내용을 설명하였다. Ⅱ장의 이론적 배경에서는 온라인 게임서버와 고가용성 시스템 그리고 Heartbeat에 대한 기술적인 내용을 다루고 있다. Ⅲ장에서는 Ⅱ장의 이론적 배경에서 얻어진 온라인 게임 서버 기술 및 고가용성 시스템 기술을 통하여 고가용성 온라인 게임 서버를 제안하고자 한다. Ⅳ장 결론에서는 논문의 내용을 요약하면서 결론을 도출하고 향후 연구 방향을 제안하였다.

Ⅱ. 온라인 게임 서버와 고가용성 시스템의 개요

1. 온라인 게임 서버

온라인 게임 시장은 급속한 발전을 거듭하고 있다. 온라인 게임의 발표가 계속이어 지고 있어 온라인 게임 사용자들은 기하급수적으로 늘어가고 있는 상황이다. 이에 온라인 게임 서버들은 많은 사용자에 따른 과부하로 인하여 속도 저하(게임에서 말하는 랙)와 심지어는 서버가 다운되는 경우까지 생겨나고 있다. 온라인 게임 서버는 사용자 증대로 인한 과부하 및 하드웨어 장애에 대응하기 위하여 다양한 기술들을 접목시켜 오고 있다. 이 장에서는 온라인 게임 서버의 일반적인 기술에 대하여 알아보기로 한다.

1) 게임 서버 기반 기술

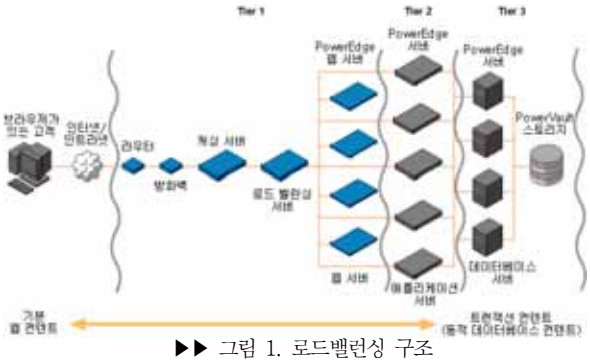
게임 서버는 온라인 게임에서 핵심적인 역할을 하는 곳이다. 모든 클라이언트의 접속을 관할하고 사용자의 데이터를 저장하는 등의 역할을 수행하는 곳이기 때문이다. 만약 게임 서버가 다운되었다고 생각해 보자. 그러면 온라인 게임의 사용자들은 접속도 하지 못할 상황에 이르게 된다. 게임 서버는 이론적으로 절대 다운이 되어서는 안 된다. 이에 서버 제작에는 높은 단가 및 안정성이 높은 하드웨어를 사용하고 높은 수준의 테크닉을 필요로 하게 된다. 여기서는 게임 서버를 제작하는데 있어 필요한 기반 기술에 대하여 알아보기로 한다.

(1) 클러스터(Cluster)

현재 컴퓨팅 파워를 증가시키기 위하여 다양한 구조의 컴퓨터가 사용되고 있다. 이 중에서 고성능 단일 컴퓨터를 이용한 계산은 이미 한계가 있음이 입증된 상태이며 그 대안으로 다수의 프로세서(CPU)를 사용하여 계산하는 병렬컴퓨터가 등장하게 되었다. 병렬컴퓨터는 다수의 프로세서가 메모리를 공유하는 SMP(Symmetric Multiprocessing)머신, 각 프로세서가 메모리를 독립적으로 가지고 있는 MPP(Massively Parallel Processing)머신, 프로세서의 지역 메모리를 계층적으로 공유하는 NUMA(Non-Uniform memory Access) 등이 있다[1]. 최근 마이크로프로세서들은 기술의 발전으로 우수한 성능을 지니고 있고 고속네트워크의 보급으로 인하여 단일 컴퓨터들을 네트워크로 연결하는 새로운 개념의 병렬컴퓨터가 만들어지는 것이 가능하게 되었다. 공개 운영 체제인 리눅스(LINUX)는 강력한 네트워크 성능을 제공하며 소스 공개로 인하여 자유로운 수정이 가능하기 때문에 클러스터를 위한 운영체제로 널리 사용되고 있다. 이러한 개념의 병렬컴퓨터를 통칭하여 클러스터라고 지칭한다.

(2) 로드밸런싱(Load Balancing)

로드밸런싱은 여러 대의 처리기가 병렬로 작업을 처리하는 다중 처리기 시스템에서 각 처리기에 걸리는 부하의 정도에 따라 한 처리기에 너무 많은 부하가 걸리거나 너무 적게 걸려 낭비되지 않도록 작업을 적절히 분배하고, 필요한 경우에는 작업을 한 처리기에서 다른 처리기로 이동시키는 것을 로드밸런싱(Load Balancing)이라고 한다.



▶▶ 그림 1. 로드밸런싱 구조

이것은 말 그래도 부하가 집중될 수 있는 부분에 균형을 잡아주는 역할을 담당하는 시스템으로써 그림에서 Load Balancer라는 것이 이러한 역할을 담당하는 시스템이고, Client는 Load Balancer 에 의해 부하가 적게 걸린 Server1로 할당받아 직접 통신을 하게 된다.

(3) 분산시스템

분산시스템이란 네트워크로 연결된 다수의 독립적인 컴퓨터들이 분산처리를 수행하는, 즉 서루 일을 분담하여 작업을 처리하는 시스템을 말한다. 이러한 분산 시스템은 서로 통신하면서 데이터나 장치들을 공유하는 여러 가지 이점이 있는 반면에 보안이나 설계부분이 상당히 어려운 점이 있다.

수천, 수만의 사용자들이 동시에 접속하는 대용량(Massive) 온라인 게임에서 넓은 세계를 구현하기 위해서는 높은 수준의

고성능 하드웨어를 필요로 한다. 분산 서버 기술은 수많은 사용자들이 동시에 접속, 안정된 시스템에서 게임을 즐기게 하기 위해 로그인과 실제 게임 세계를 분리하는 방법을 채택, 비교적 저렴한 하드웨어로도 이런 기능들이 충분히 수행될 수 있도록 구현하게 해 준다.

분산 서버 기술은 접속을 관장하는 한 대의 로그인 서버와 게임내의 각 지역에 존재하는 사용자 데이터를 처리하는 많은 로컬 서버들이 스타(Star)구조로 연결되어 한 서버의 적정 수용인원을 5~600명으로 유지하고 있다. 이러한 기술은 게임의 확장성을 넓혀주는 중요한 요소로 작용한다.

2) 게임 서버 제작 형태

온라인 게임은 대부분 클라이언트 - 서버 구조를 가지고 있다. 이에 온라인 게임에서의 게임 서버는 가장 중요한 역할을 하기 때문에 초기 게임 서버의 설계단계에서는 효율성, 확장성 그리고 안정성 등의 여러 가지 사항을 고려하게 된다. 초창기의 게임 서버는 중앙 집중식 게임 서버를 채택하였지만 현재는 게임의 규모가 커지고 게임 사용자 또한 많아짐에 따라 중앙 집중식 게임 서버로는 도저히 감당하지 못할 정도가 되었다. 그리하여 중앙 집중식 보다는 분산 서버 구조로 변하게 되었다. 여기에서는 여러 가지 종류의 게임 서버에 대한 특징을 알아보고자 한다.

(1) 중앙 집중식 서버 구조

중앙 집중식 또는 단일 서버 구조라 불리는 가장 기본적인 서버 구성 기법으로 한 대의 게임 서버에 사용자들이 접속을 하여 서비스가 이루어지는 구조이다. 비교적 작은 규모의 서비스에 적합한 구성으로 서버의 구성비용이 적게 들고 구현이 쉽고 운영에 있어서 쉬운 장점이 있다. 반면 서버가 하나뿐이라는 것 때문에 수용 가능한 클라이언트의 수가 한정될 수밖에 없어 서비스에 지장을 줄 우려가 있고 확장성 또한 용이하지 않다. 이러한 단점을 극복하기 위하여 고성능의 HPC(High Performance Computer)를 사용할 수 있으나 도입 비용이 많이 들고 고성능이라 하지만 사용자의 한계가 있다는 단점이 있다. 또한 서버가 다운되게 되면 모든 서비스가 중단되어야하는 상황이 발생하기 때문에 대규모의 서비스에는 적합하지 않은 구조이다.

(2) 분산 게임 서버 구조

온라인 게임을 즐기는 사용자들이 급격히 증가하면서 중앙 집중식 서버로 운영되는 게임 시스템은 한계에 도달하게 되었다. 그 이유는 게임 사용자의 증가로 인하여 서비스 속도는 느려지고 다운 현상도 발생하는 등 서비스 질이 급격히 떨어졌기 때문이다. 따라서 대규모의 온라인 게임에서는 다중 서버 구조를 사용하는 분산 구조 게임 서버를 채택하고 있다. 분산 구조 게임 서버는 말 그대로 각 서버간의 부하를 일련의 프로세스를 통하여 분산시켜 주는 방식이다. 똑같은 게임 서버와 DB서버를 여러 개로 구성하여 사용자가 증가할 경우, 그에 따라 서버의 수도 증가시키는 구조이다. 이러한 형태는 초기의 온라인 게임이 나오면서 개발된 시스템으로써, 중앙 집중식 서버 구조에 비해서 많은 장점을 가지고 있다. 큰 장점이라면 온라인 게임 같은 경우는 많은 사용자들이 하나의 서버를 통해서 게임을 하기 때문에 서버가 중단되면 치명적인 영향을 미치게 된다. 그래서 이러한 서버의 다운 현상으로부터 서비스가 중단되는 사태를 막을 수 있는 구조가 바로 분산 구조 서버이다. 하지만 단점으로는 초기의 시스템 구성비용이 2배 이상으로 소요된다.

2. 고가용성 시스템

1) 고가용성 시스템

고가용성(High-Availability) 서버는 하드웨어나 소프트웨어 또는 네트워크 등의 자원에 결함이 발생하였을 경우라도 지속적으로 서비스를 제공할 수 있어야 한다. 결함(Failure)과 시스템의 다운타임(Down Time)을 최소화함으로써 서비스의 손실을 줄일 수 있도록 컴퓨터 시스템을 구축하고 관리하는 시스템을 말한다. 서비스 중단은 크게 계획된 중단(Planned Down)과 계획되지 않은 중단(Unplanned Down)으로 나눌 수 있다[2]. 계획된 중단은 백업(Backup), 업그레이드(Upgrade), 유지보수(Maintenance) 등 계획에 의해 서비스가 중단되는 경우이고, 계획되지 않은 서비스 중단은 정전, 하드웨어 또는 소프트웨어 결함, 네트워크 장애, 물리적인 시스템 파괴, 자연 재해 등에 의해 서비스가 중단되는 경우이다. 하나 또는 그 이상의 요소에 의하여 서비스 전체가 중단되는 하드웨어 또는 소프트웨어 요수를 SPOF(Single Point Of Failure)라 한다[2]. 하드웨어적인 SPOF로는 CPU, HDD, Main board, Memory, Ethernet Card등이 SPOF라 할 수 있다. 이러한 하드웨어적인 SPOF에 의한 서비스 중단은 해당 SPOF의 중복적인 구성으로 이러한 문제를 해결할 수 있다. 따라서 고가용성 서버는 물리적인 중복 구성으로 하드웨어적인 SPOF에 의한 서비스 중단을 회피할 수 있으며, 소프트웨어 기법을 통하여 소프트웨어적인 SPOF에 의한 서비스 중단을 방지할 수 있다. 물리적인 중복 구성은 서비스를 제공하는 서비스 노드(Service Node)와 서비스 노드의 결함 발생 시 서비스 노드 대신에 지속적으로 서비스를 제공할 수 있는 백업 노드(Backup Node)로 구성됨을 의미한다. 즉, 고가용성 서버는 서비스 노드와 백업 노드로 이루어진 고가용성 클러스터(Cluster)서버이다. 고가용성 클러스터 서버는 결함이 발생한

서비스 노드에서 대기 중인 백업 노드로 제공 중인 서비스를 빠르게 전환하여 서비스 중단을 최소화하고 사용자에게 서비스를 지속적으로 제공한다.

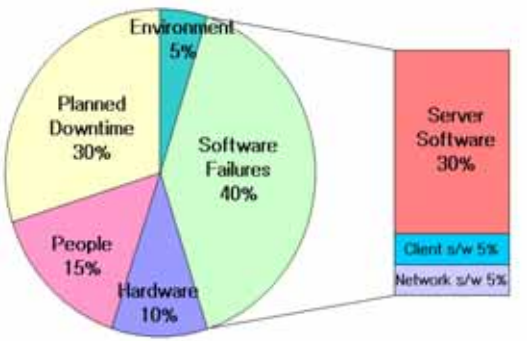
2) Heartbeat

고가용성 시스템은 어떠한 문제가 발생하더라도 서비스를 지속적으로 수행할 수 있어야 한다. 그러기 위해서는 현재 서비스를 수행하고 있는 노드가 지속적으로 작업을 수행할 수 있는지에 대하여 주기적으로 확인하여야 한다. 이때에 쓰이는 기술이 Heartbeat이다. Heartbeat은 고가용성 시스템내의 결함 여부를 상호간에 체크하여 필요한 조치를 취하도록 한다[3][4].

Heartbeat에는 시스템 점검 기능과 IP takeover기능이 있으며, Heartbeat 메시지는 시리얼라인, UDP, PPP/UDP 등을 통하여 전송될 수 있다[4]. Heartbeat 전송 후 상대 노드가 정상적이면 응답을 하지만 만약 응답 메시지가 정해진 시간이내에 도착되지 않으면, timeout 카운트가 시작되고 계속해서 정해진 개수만큼 heartbeat 메시지에 대하여 응답 메시지를 받지 못하면 상대 노드는 장애가 발생한 것으로 간주한다[5]. 각 노드들은 주기적으로 heartbeat를 주고받으며 서로의 상태를 확인한다. 이때 서비스를 수행 중이던 노드가 결함이 발생하였다고 판단되면 정상적인 노드(또는 대기 중인 노드)가 서비스를 수행한다. High-Availability LINUX Project의 heartbeat 패키지는 configuration파일을 통하여 heartbeat 간격을 사용자가 입력하도록 되어 있다. 그러나 heartbeat 간격을 좁게 하면 시스템 결함 발생 시 신속히 결함을 인지할 수 있으므로 태스크 수행손실을 적게 할 수 있지만 heartbeat 비용이 크게 되고, 반대로 heartbeat 간격을 크게 하면 heartbeat 비용은 적지만 시스템 결함 발생 시 결함 인지 시간이 길어짐에 따라 태스크의 재수행 시간이 크게 지연되므로 최적의 heartbeat 간격에 대하여 고려하여야 한다.

Ⅲ. HeartBeat를 이용한 고가용성(HA) 보장 온라인 게임 서버 구축 제안

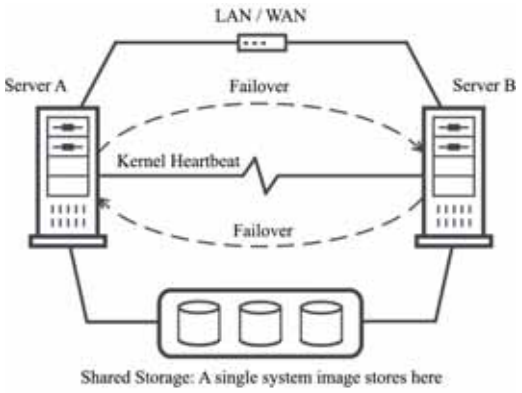
온라인 게임 서비스에서 가장 중요한 부분은 게임 서버의 지속적인 서비스 능력이다. 게임 서비스의 지속성이야말로 게임 서비스에서 가장 중요한 부분이라고 생각된다. 앞에서 말한바와 같이 게이머가 게임을 즐기고 있는 동안 게임 서버에는 많은 양의 데이터가 네트워크를 통해 클라이언트와 교환을 하고 있다. 이러한 상태에서 서버의 장애가 발생하면 서버와 클라이언트간의 교환중인 데이터의 손실 및 게임 서비스에 많은 장애를 가져오게 된다. 그림 2에서와 같이 서비스 중단의 원인 중 40%를 차지하고 있는 것이 소프트웨어 결함이고 다음으로 10%를 차지하는 것이 하드웨어의 결함이다. 이러한 여러 가지 게임 서버의 장애(네트워크 장애, 게임 서버 하드웨어의 고장, 소프트웨어 고장 등)로 인한 서비스의 차질을 줄이기 위하여 다음과 같은 게임 서버 시스템을 제안하고자 한다.



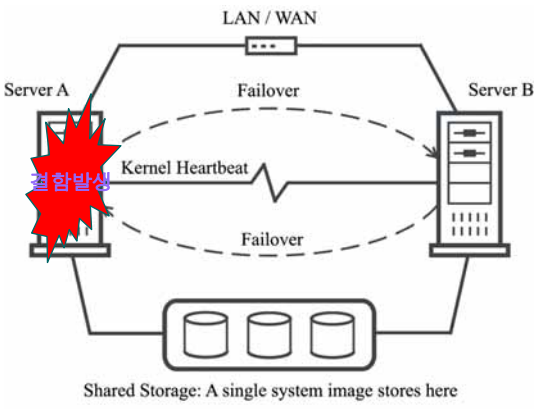
▶▶ 그림 2. 서비스 중단의 원인

1. 제안 시스템 구성

이 논문에서 제안하고자 하는 시스템은 데이터베이스를 공유하는 두 개의 복제된 게임 서버를 두고 이를 통하여 고가용성 시스템을 구축하고자 한다. 고가용성 시스템 구성을 위하여 heartbeat 기술을 이용하고자 한다.



▶▶ 그림 3. 게임 서버 시스템 제안



▶▶ 그림 4. 제안 시스템 구동 방식 예시

그림과 같이 두 개의 게임 서버와 공유 데이터베이스를 구성한다. TCP/UDP 네트워크로 연결됨으로써 서비스를 실행하게 된다. 여기서 게임 서버(A)와 게임 서버(B)는 UDP와 함께 시리얼 케이블 일명 FX케이블을 통하여 연결되어 있다. 네트워크 및 시리얼케이블로 연결된 게임 서버 A와 B는 heartbeat을 통하여 결함유무를 확인한다. 게임 서버A, B 는 네트워크와 시리얼 케이블로 연결되어 있고, 컴퓨터들은 각각 2개의 NIC를 가지고 있어서 외부 네트워크(public network)와 내부 네트워크(private network)로 연결되어 있다. 내부 네트워크와 시리얼 케이블은 노드들 간의 heartbeat 통신을 위해서 사용된다. 게임 서버A, B는 주기적으로 heartbeat 신호를 통해 다른 컴퓨터의 상태를 체크해서 서비스가 이루어지고 있는지 확인한다. 컴퓨터의 NIC도 SPOF가 될 수 있기 때문에 heartbeat는 내부 네트워크와 시리얼 케이블을 통해 다른 컴퓨터가 살아있는지 여부를 판단한다. 만약 컴퓨터 노드들 간에는 서로를 감시하기 위해서 시리얼 케이블로 연결되어 있는데 각 노드가 모든 노드를 연결하는 것이 가장 확실한 방법이지만 이러한 경우에 N 개의 노드를 사용하면 N*(N-1) 개의 시리얼 라인이 필요하기 때문에 확장성이 떨어진다.

2. 제안 시스템 구동 방식

게임서버 A, B는 주기적으로 HEART_BEAT_PACKET을 전송하여 게임 서버 A, B의 상태를 체크한다. 만약 어느 일정한 시간동안에 상대방에게서 응답이 없을 경우에는 일시적인 장애나 시스템 부하일 경우에 대비하여 미리 정해진 횟수만큼 다시 패킷을 보내고, 패킷이 돌아오는 것을 기다린다. 여기서 계속적으로 응답이 없을 경우에는 게임서버 A가 중지된 것으로 간주하고 Fail-over[6]작업을 수행한다. 그림5는 게임 서버 A,B사이의 heartbeat 신호를 체크하는 알고리즘을 나타낸다.

게임 서버A가 중지되면, 게임 서버 B에 장애 사실을 브로드케스트하고, 동적으로 다시 정상적으로 수행되는 게임 서버B가게임 서버A를 대신하여 동작한다. 그림6은 fail-over 알고리즘을 나타낸다.

```
Heartbeat algorithm
begin
    while( forever )
        send HEART_BEAT_PACKET to neighbor
        receive HEART_BEAT_PACKET from neighbor
        if( don't receive HEART_BEAT_PACKET for a fixed time )
            send HEART_BEAT_PACKET to neighbor
            wait for a fixed time to receive HEART_BEAT_PACKET
            if( don't receive HEART_BEAT_PACKET for a fixed time )
                start fail-over
            end if
        end if
    end while
end
```

▶▶ 그림 5. heartbeat 신호 점검 알고리즘

```
Fail-over algorithm
begin
    broadcast the system fault information to all nodes in the cluster
    rebuild cluster dynamically with active nodes
    if( this node == the first node in the cluster )
        mark this node as controller
        for( all nodes in the cluster )
            send SYSTEM_LOAD_PACKET
            receive SYSTEM_LOAD_PACKET
        end for
        find out the node with the lowest system load
        send FAIL_OVER_PACKET to the node
    else
        receive SYSTEM_LOAD_PACKET
        check system load
        send SYSTEM_LOAD_PACKET to the controller
    end if
    if( this node == the node with the lowest system load )
        start fake_process
        setup system for take-over services
        start services
    end if
end
```

▶▶ 그림 6. fail-over 알고리즘

fail-over가 발생한 후에 다운되었던 게임 서버 A가 다시 정상적으로 복구되면, fail-back[6] 작업을 위한 적절한 작업이 필요하다. fail-back 작업은 다음과 같이 이루어진다. 첫 번째, 다운되었던 게임 서버 A를 재부팅한다. 재부팅한 게임 서버 A는 내부 IP를 가지고 있다.

게임 서버 B에 IP_CHECK_PACKET을 전달하고, 이 신호를 받은 게임 서버 B는 게임 서버 A의 서비스를 수행하고 있는지 여부를 CHECK_IP_ANSWER_PACKET을 통해 전달한다. 게임 서버 B가 게임 서버 A의 서비스를 수행하고 있기 때문에 게임 서버 B에 FAIL_BACK패킷을 전달한다. FAIL_BACK패킷을 받은 게임 서버 B는 게임 서버 A에 서비스를 넘겨주기 위한 작업을 수행하고, 작업이 끝나는 경우에 RESPONSE를 게임 서버 A에 전달한다. 이 RESPONSE를 받은 게임 서버 A는 외부 IP를 활성화시키고, 서비스를 제공한다. 그림 7은 위에서 설명한 FAIL_BACK 알고리즘 기술을 나타낸 것이다.

```
Fail-back algorithm
begin
    boot the system with a internal IP
    check Diehard configuration
    for( all nodes in the cluster )
        send CHECK_IP_PACKET
        receive CHECK_IP_ANSWER_PACKET
        find out take-over server
    end for
    if( take-over server exists )
        send FAIL_BACK_PACKET
        receive FAIL_BACK_OK_PACKET
    end if
    setup external IP
    check computing resources for services
    start services
end
```

▶▶ 그림 7. fail-back 알고리즘

3. 제안 시스템의 장점

상업적으로 서비스를 실시하고 있는 시스템에서의 서비스 중단은 업체에게 심각한 손실을 초래할 수 있다. 1995년 오라클과 데이터메이션의 보고서에 의하면 갑작스러운 서비스 중단은 평균적으로 시간당 8,000~35,000달러의 경제적 손실을 가져온다[7]. 표1에서 보듯이 최근 데이터에 의하면 서비스 중단에 따른 손실은 더욱 증가하고 있다. 표1은 서비스 중단에 따른 업체별 시간당 손실액을 보여준다[8].

[표 1] 서비스 실패에 따른 비용

업 체	시간당 비용
중개 사업	$6.45M
신용 카드/판매 인증	$2.6M
홈 쇼핑(TV)	$113K
홈 카달로그 판매	$90K
항공 예약	$89.5K
전화 티켓 판매	$69K
패키지 해운업	$28K
ATM 수수료	$14.5K

게임 서비스에서도 마찬가지이다. 서비스 도중 발생하는 결함으로 인하여 게임 데이터의 손실과 함께 게이머로부터의 많은 컴플레인(complain)을 받는 것이 사실이다. 이러한 경우 게임 업체의 신뢰도 및 게임 서버의 서비스 지속성에 영향을 줌으로써 향후에는 그 게임 업체의 차후 게임의 완성도 및 사용자들의 신뢰를 무너뜨릴 수도 있을 것이다. 이러한 신뢰도를 더 높이기 위해서는 본 논문에서 제안하는 고가용성 게임 서버 시스템을 도입함으로써 해결 가능하다. 모든 게임 서버에서 각각의 대응 standby server를 두어 항상 예기치 못하게 일어나는 real server의 문제점(하드웨어 장애, 네트워크 장애, 프로그램장애)으로 인한 서비스 장애에 대하여 standby server가 대신 서비스를 수행함으로써 장애 시간을 대폭 줄일 수 있기에 게임 서비스의 신뢰도 향상을 기대할 수 있다.

또한, 요즘의 온라인 게임은 커뮤니티의 활성화로 인하여 게임의 활성화가 이루어지고 있다고 할 수 있다. 게임의 길드화를 통하여 커뮤니티를 형성하고 온라인상에서 같이 게임을 즐김으로써 게임의 재미를 배가시킨다. 길드를 통하여 커뮤니티화된 게임의 유지에 본 논문에서 제안하는 시스템은 다음과 같은 장잠을 가지고 있다. 블리자드사의 스타크래프트(Starcraft) 게임을 예로 들어보고자 한다. 스타크래프트에서 온라인 대전을 하기 위하여는 배틀넷(Battlenet)이라는 게임 서버에 접속을 하여야 한다. 블리자드의 배틀넷 방식은 여러 개의 지역서버를 두어 접속하는 것이다. 아시아 서버를 주요 활동 무대로 하는 길드가 있다고 하자. 만약 배틀넷의 아시아 서버가 문제가 생겨 서비스의 장애가 있으면 아시아 서버에는 더 이상 게임 사용자의 접속이 불가능하다. 그런 상태라면 아시아 서버를 주 무대로 활동하고 있는 길드라면 다른 유럽 서버나 미국 서버에 접속하여 게임을 개인적으로 즐길 수는 있지만 길드를 통한 활동을 하지 못한다. 모든 길드 정보가 아시아 자역서버에 저장되어 있기 때문이다. 이에 Standby server가 존재한다면 이러한 문제는 한 순간에 해결된다. 서비스의 장애로 인한 접속 불능 타임도 현저히 줄어들 것이다.

Ⅳ. 결 론

본 논문에서는 게임 서버에서의 고가용성 시스템 구축을 위하여 heartbeat 알고리즘의 사용을 제안하였다.

게임 서비스와 같은 네트워크 기반의 시스템에서 사용자는 시스템의 장애, 네트워크 혹은 서버 부하에 의한 서비스 품질의 저하를 구별하지 못하고 또한 기다리는 동안을 일종의 서비스 실패라고 취급한다[9]. 이러한 서비스 실패를 줄이고 게임 서버의 가용성을 90%이상으로 올리기 위하여는 heartbeat알고리즘을 통한 고가용성 게임 서버를 구축함으로써 서비스 실패를 줄이고 게임 커뮤니티를 더욱 활성화할 수 있을 것이다. 이에 본 논문에서 제안한 시스템을 직접 구현하여 실제적인 입증을 하려 한다.

▌참 고 문 헌▐

[1] 배재환, 부하분산 게임서버를 위한 리눅스 웹 클러스터 시스템 설계 및 구현, 한국컴퓨터게임학회 논문지, 제1권 1호, pp.2-85, 2002.
[2] Harald Milz, "LINUX High Availability HOWTO", http://halinux.rug.ac.be/High-Availability-HOWTO-5.html, Dec. 1998.
[3] Tom Vogt, "Heart redundant, distribute cluster technology", http://www.lemuria.org/Heart, Sep.1999.
[4] Alan Robertson, "LINUX-HA Heartbeat System design", The 2000 ASL conference, Oct 2000.
[5] 최재영 외, "고가용성 리눅스", 한국정보처리학회지, 제6권 6호, pp.19-25, jun 1999.
[6] Gregory F. Pfister, In search of clusters, Prentice Hall PTR, 1998.
[7] Linux HA HOWTO, http://metalab.unc.edu/pub/Linux/ALPHA/linux-ha/High-Avaliability-HOTWTO.html
[8] Evan marcus, Hal Stern, Blueprints for hign availability, Wiley & Sons, 200.
[9] Alan Wood, "Predicting Client/Server Availability", in IEEE Computer, pp.41-48, Aprill 1995.